Table of Contents

Contract No.: 53-3198-0-22
MPR Reference No.: 7925-050

MICROSIMULATION ON PERSONAL COMPUTERS: A FEASIBILITY STUDY

October 2, 1992

Authors:

Robert Cohen
Harold Beebout
Pat Doyle
Julie Sykes

Prepared for:

U.S. Department of Agriculture
Food and Nutrition Service
3101 Park Center Drive
Alexandria, VA 22302

Project Officer:
Alana Landey

Prepared by:

Mathematica Policy Research, Inc.
600 Maryland Avenue, S.W.
Suite 550
Washington, D.C. 20024

Project Director:
Harold Beebout

This work was prepared as one task of a competitively awarded contract; the total amount of the contract is $2,854,698.

CONTENTS

Table of Contents

CONTENTS (continued)

CONTENTS (continued)

TABLES

FIGURES

TRADENAMES

The following tradenames are used throughout this report.

TRADE NAME	MANUFACTURER
UNIX - AT&T	AT&T
IBM PC PC-XT PC-AT OS/2	IBM
PDP VAX	Digital Equipment Corporation
MS-DOS Visual Basic Windows	Microsoft Corporation
POSIX	Institute of Electrical and Electronics Engineers
8088 80286 80386 80486	Intel Corporation
Bernoulli	IOMEGA Corporation
SAS	SAS Institute, Inc.
VISICALC	Visicorp, Inc.
Wordstar	Wordstar USA
NORTON	Symantec Corporation
TPL	QQQ Software
dBASE	Borland International, Inc.
CLIPPER	Nantucket
FASTBACK PLUS	Fifth Generation Systems
MATH	Mathematica Policy Research, Inc.
TRIM TRIM2	The Urban Institute
SPSD/M	Statistics Canada
CORSIM	Steve Caldwell/Cornell University

I. INTRODUCTION

At the request of the Assistant Secretary for Planning and Evaluation, Department of Health and Human Services (ASPE), and the Food and Nutrition Service (FNS), U.S. Department of Agriculture, the Committee on National Statistics of the Commission on Behavioral and Social Sciences and Education of the National Research Council convened a panel in 1988 to evaluate microsimulation models for social welfare programs. In the words of the panel, "We are excited about the prospect that powerful new hardware and software technology will make possible a new generation of microsimulation models that support increased modeling capabilities and flexibility for analysts" (Citro and Hanushek, 1991, p. 191). The panel went on to recommend that agencies, such as FNS and ASPE, that are responsible for developing public policy support the movement of microsimulation models from mainframe computers to the powerful new personal computers (PCs) or workstations. The panel was very encouraging about the potential for PC-based microsimulation models to assist in achieving the objectives embodied in other panel recommendations. These objectives include:

- Facilitating access to models by policy analysts and others who are not computer systems experts
- Reducing the time, effort, and computing cost of preparing policy simulations
- Adding information on the level and sources of uncertainty in the simulation estimates
- Improving the documentation on the models and the procedures for archiving earlier simulations and versions of the models

The panel further recommended that efforts to shift microsimulation models to PCs proceed in a incremental or staged manner. This would allow the lessons learned in one stage to be incorporated in the development effort before moving to the next stage of model improvement.

This report has a dual purpose. First, it documents the plan of Mathematica Policy Research, Inc. (MPR) for moving a set of components of the Micro-Analysis of Transfers to Households (MATH®) model routinely used in responding to FNS requests for analysis of the Food Stamp Program (FSP) to the PC environment. Second, it describes more generally our overall design goals for moving the entire MATH model to the PC environment, emphasizing our long-term goals and presenting an explicit plan for the next stage of the work. In Chapter II, we provide an overview of both microtechnology and microsimulation models other than MATH that operate on the PC. Chapter III explains the functions, development, design, and operating environment of the current MATH model. Chapter IV presents the design goals for the proposed PC-based MATH model. Chapter V presents the results of testing the PC to determine whether it is capable of handling the MATH model, and Chapter VI explains our approach to implementing the model on the PC.

II. OVERVIEW: MICROCOMPUTER TECHNOLOGY AND MICROSIMULATION ON THE PC

Microcomputers have dramatically affected the way in which we work in almost every field, including microsimulation. In this chapter, we discuss the state of the art in microcomputer technology, and we describe several microsimulation models that operate on the PC.

A. THE STATE OF THE ART IN MICROCOMPUTER TECHNOLOGY

Sustained rapid advances in computing capabilities continue to shift much of our quantitative research operations away from mainframe systems to the microcomputer. This section documents several of the most important advances in processing speed and disk speed, and in increased storage capacity. It also explains the declining cost of microcomputers. To provide a context for this discussion, we define microcomputer architecture and present an overview of the evolution of the PC. Because this report depends on technical language, we have included a glossary of terms in Appendix A.

1. Microcomputer Architecture

There are four major components of a microcomputer. The central processing unit (CPU), or chip, is the heart of the computer.[1] It is where all the program instructions are carried out.[2] The second major component is the memory (random access memory, or RAM). This is where the program temporarily resides as it is being executed. The third component is permanent storage, which may take the form of hard disks or floppies. The fourth component comprises the video, keyboard, and printer subsystems. The overall performance of the microcomputer depends on the first three components; it would be pointless to substantially increase the capacity of one component

[1]In this report, the terms CPU, chip, microcomputer processor, and microprocessor are used interchangeably.

[2]A program is a set of instructions that tell the computer to perform a task. A program can perform tasks as simple as displaying "hello world" or as complex as running the MATH model.

without correspondingly increasing the other components. We focus on the development of the first three components in this report, since the last component does not significantly affect model processing.

2. Evolution of the PC

By today's standards, the original IBM PC was a very small machine with a mere 64,000 characters or 64 kilobytes (KB) of memory and a single-sided floppy drive with 160KB of storage. Introduced in 1981, the IBM PC used an off-the-shelf microcomputer processor from Intel, designated the 8088. The processor had a 16-bit CPU with an 8-bit data path (Table II.1). It was a stunningly powerful computer for its time.

TABLE II.1

DEVELOPMENT OF THE INTEL CHIP FOR THE IBM PC FROM 1981 TO 1993

Chip	First Full Year of Production	Word Size[b]	Memory Addressability	Instructions Per Second	Typical Clock Cycles Per Second
8088	1981	8/16	1MB	300K	5MHz
80286	1985	16	16MB	1.5M	12MHz
80386	1987	32	4GB	5M	25MHz
80486	1990	32	4GB	15M	40Mhz
80586[a]	1993	32/64	N/A	100M	100Mhz

SOURCE: Henke and Kuvshinoff (1992).
[a]Not introduced yet; these are estimates.
[b]A key measure of a machine's capacity, word size is the number of bits the chip typically transmits. Larger values reflect more powerful chips.
N/A: Not available.

A year or two later, IBM brought out the PC-XT, which could have up to 640KB of memory and a 10 million-byte [or 1-megabyte (MB)] hard disk. Many people felt that it was inconceivable that

anyone could harness that much power. Nonetheless, in 1984 IBM introduced an even more powerful PC, the IBM PC-AT, which became an instant success. This class of machine enabled users to quickly perform such tasks as word processing and spreadsheet calculations; however, these machines were far less successful at such applications as SAS or FORTRAN, which require much greater resources--especially in storage and computational power.

Not very long after the PC-AT made its appearance, Intel introduced the 80386 (386) chip. This new CPU opened the door to almost mainframe-like power. The hardware was capable of multitasking (running two or more programs simultaneously), of performing full 32-bit arithmetic using a 32-bit data path (the same size that current IBM mainframes use), and of addressing up to 4 billion bytes (4 gigabytes, or 4GB) of memory. At this point, the line between microcomputers, minicomputers, and even mainframes started to blur.

Continuing to move forward, Intel introduced the 80486 (486) chip in 1989. The 486 was very similar to the 80386 except that it had a built-in math coprocessor[3] and a built-in 8KB cache (which provides quicker access to data). Intel is currently working on the next generation chip. Unofficially dubbed the 80586, or P5, this chip is expected to be four or five times faster than the 80486. It should be in production in early 1993.

3. Increase in CPU Speed and Disk Speed

The most significant factors that affect the speed of a microcomputer for microsimulation programs are the CPU speed (processing speed) and the hard disk speed.

a. CPU Speed

CPU speed is often measured in relation to the speed of the original PC-XT, and we will use this as our principal measure. The frequency, or number of clock cycles, at which the chip operates

[3]A math coprocessor performs floating point or real arithmetic (arithmetic that involves numbers with decimal points) much faster than the CPU can by itself. Since much of the MATH model's work involves floating point numbers, a math coprocessor is critical to fast calculations. The cache provides quicker access to data stored in memory.

is another measure of speed, but it is useful only when measuring the speed of chips in the same class. Frequency is usually expressed in millions of cycles per second (megahertz, or MHz). We define this measure because the reader will often see mention of a 33MHz 486 or a 50MHz 486. The number of instructions in millions per second (MIPS) is another measure of speed that is often used. However, there are many types of instructions that take different amounts of time to execute, and some types of CPUs can execute several instructions at once, so this is a fairly vague measure.

Each new generation of CPU or chip has doubled or even quadrupled the processing speed over that of the previous generation. In addition, the frequency (MHz) within each CPU generation increases with newer versions of the same chip. For example, the 486 started out at 25MHz, increased to 33MHz, and the latest versions of the 486 now run at 50MHz. The frequency is linearly related to the processing power within the same class of machine. According to The NORTON Utilities, a commonly used measure of CPU speed, a 33MHz 486 PC is 70 times faster than the original PC-XT and about 33 percent faster than a 25MHz 486. The 50MHz 486 is about 50 percent faster than the 33MHz 486, or about 100 times as fast as the PC-XT. See Table II.2 for typical speeds for each type of CPU.

TABLE II.2

TYPICAL CPU SPEEDS

CPU	Relative Speed[a]	MIPS	Clock Speed
8088	1	0.25	4.77MHz
80286	4	1	8MHz
80386	33	7	33MHz
80486 25MHz	50	10	25MHz
80486 33MHz	70	14	33MHz
80486 50MHz	100	21	50MHz

[a]Relative speed measured using the SI command from The NORTON Utilities V4.5.

b. Hard Disk Speed

Two factors that affect disk speed are (1) average access time and (2) data transfer rate (DTR). Average access time is often critical to applications that read or write data randomly. Access time is generally measured in thousandths of seconds or milliseconds (ms). The MATH model reads and writes data sequentially, whereas databases use random access. The DTR is the more critical factor for microsimulation models or other applications that sequentially process large data files. Transfer rates are measured in either millions of bytes (megabytes) or in millions of bits (megabits) per second. While disks are mechanical in nature, the CPU is electronic. As a result, we have not observed the same amount of increase in speed for hard disks as has been observed for the CPU, since mechanical parts are more difficult to improve. The average access time has decreased from over 100 ms on the original PC-XT to under 10 ms on some newer hard disks. Notice that this change lags behind the relative increase in the CPU speeds. The DTR has increased from under 100,000 bytes per second to over 1MB per second. While rates of transfer achieved in real world applications will vary somewhat from these benchmark rates, the latter are indicative of the actual increase in the DTR.

4. Increased Storage Capacity

Three types of storage capacity are relevant to this report: (1) hard disk capacity, (2) memory capacity, and (3) backup capacity. The typical storage capacity of a moderately priced hard disk drive has increased from 10MB to 200MB. However, drives as large as 2 billion characters (2 gigabytes, or 2GB) have recently been announced by manufacturers of hard drives. This increase in storage capacity is essentially a response to storage requirements, which have risen with each new generation of software. We benefit from more storage capacity, since a PC-based MATH model will require substantial disk space.

As mentioned, memory is where a program and its data are stored while it is being executed. This memory is often called RAM for random access memory or DRAM for dynamic random access memory. The typical amount of RAM installed on a PC has risen from 64KB in the original PC to

more than 4MB, which is the minimum required to run many current software packages and operating systems such as Windows or OS/2. Many PCs are capable of holding 32 or 64MB of memory (or 1,000 times the capacity of the original PC).

Backup capacity refers to the ability to preserve data on a secondary medium to prevent data loss. Backup or archival mechanisms for the PC have evolved from diskettes to 1/4-inch tapes to digital audio tape (DAT) and optical disk storage. Floppies were originally single sided with 160KB of storage. They are now double-sided and have evolved to 1.2MB, 1.44MB, 2.88MB, and more recently to 20MB sizes. Backup tape drives, which started out with a capacity of 20MB, have grown to 700MB or more for 1/4-inch tapes, 2.2GB for 4-mm DATs without compression, and 8-mm DATs that hold 5GB without compression. Compression (which refers to reducing the size of a data set through eliminating redundancy) can double these storage capacities.

Magnetic optical (MO) disk technology is the latest storage medium. MO has some advantages over the other technologies, since (like the Bernoulli drive technology that MPR uses extensively) the system treats it like another hard disk. Therefore, the user does not have to learn new commands or procedures. The typical sizes for optical disks are 128MB, 600MB, and 1GB.

5. Evolution of Software

Without applications or software, a computer is just a collection of metal, plastic, and silicon. The PC was not widely useful until programs like VISICALC (the first spreadsheet) and Wordstar (very close to the first word processor for a PC) were introduced. The availability, capability, and sophistication of software has grown tremendously in the past decade, although not nearly as fast as the hardware has advanced. Software is just now beginning to catch up to the hardware capability introduced with the 80386. An operating system (OS) is the framework on which applications rely to load and execute; to provide printer, video, and disk services; and to interface in general with the system. Operating systems like OS/2 and UNIX are able to use most of the power of today's technology, unlike DOS, which is designed to run on 8088s and up. Since DOS is written for the

least common denominator, it is not able to use many of the enhancements of newer chips and thus suffers performance loss. Nonetheless, some versions of the FORTRAN and C programming languages are able to make use of the full range of memory and storage provided by current hardware.

The most popular operating system used is Microsoft's MS-DOS. MPR and most government agencies use this system for most of their day-to-day work. However, both IBM and Microsoft are developing competing operating systems. This makes it difficult to predict the future of OS, but we believe that in the long run, this competition will help the user. IBM is developing OS/2 2.0, and Microsoft is developing WINDOWS NT. Both offer a great deal more power than DOS, but we are more concerned with OS compatibility with software packages than with OS power, per se. Windows NT is not currently available, but OS/2 2.0 is. We have done some limited testing on it, and we find it robust and fairly fast for our applications. It offers a high-performance file system and a full 32-bit development environment. Both of these features offer improvements in speed and performance over DOS. We expect similar results from Windows NT.

6. Declining Costs

Another important measure of the change in the PC computing environment is the fact that the cost of the typical PC has dropped almost as drastically as the power has increased. The net effect of this is made clear in Table II.3. In 1985, one could buy an IBM PC-AT for approximately $6,000. This money purchased a 6MHz 80286, a 20MB hard disk, a monochrome monitor, a math coprocessor, a 1200-baud modem, and an 80-character-per-second dot matrix printer. In current dollars, $6,000 now buys a very fast 50MHz 486 computer with a quality color monitor, 8MB of memory, 1GB hard disk drive, and a laser printer. This computer can outperform the original AT by a factor of 20 and the PC-XT by a factor of 100 or more for most types of applications. In addition, the per-megabyte cost for storage media has dropped 100-fold in the past decade. (See Tables II.4 and II.5 for some comparisons.)

TABLE II.3

COMPARISON OF PC PRICES OVER TIME

Computer	Typical Drive	Original Package Cost	Current Price
8088 XT	10MB	$6,000 in 1982	$600
80286 AT	20MB	$6,000 in 1985	$700
80386	40MB	$6,000 in 1987	$1,800
80486	100MB	$6,000 in 1989	$2,000

TABLE II.4

COST PER MEGABYTE OF STORAGE FOR HARD DISKS OVER TIME

Year	$/MB	Typical Capacity
1982	300.00	10MB
1984	60.00	20MB
1992	3.00	100MB

TABLE II.5

CURRENT COST PER MEGABYTE FOR DIFFERENT STORAGE MEDIA

Media	$/MB	Typical Capacity
Hard Disk	3.00	100 - 1000MB
Tape		
¼-inch	0.10	20 - 250MB
DAT	0.01	1000 - 2200MB
Bernoulli	1.50	44 - 90MB
Rewritable Optical	0.30	128 - 950MB

The cost/benefit ratio derived from PCs with current technology has opened a vast range of opportunities for downsizing applications from mainframes to PCs. The technical advances discussed in this chapter allow us to seriously consider moving the MATH model from the mainframe to the PC. Most barriers have been removed, and there is no end in sight to the technological explosion that started ten years ago. We can be certain that tomorrow's computer will have capabilities of which we have not yet dreamed.

B. MICROSIMULATION APPLICATIONS ON THE PC

PC-based microsimulation models were rare until very recently. In this section, we describe models with objectives similar to those of the MATH model that are operating effectively on the PC.

1. QC Minimodel

Data on the sample of Food Stamp Program (FSP) cases drawn for quality control review purposes have been used since the mid-1970s for simulating the impacts of program changes on current program participants. Until 1987-88, these simulations were done by FNS or MPR staff using SAS on a mainframe computer. Starting in 1987, a microsimulation model was developed by MPR for the PC environment. The model was originally implemented in PC SAS. SAS proved to be too slow and limiting for the quick-response demands, and the processing portion of the model has since been converted to FORTRAN. The QC Minimodel makes extensive use of menus to (1) define program reforms and the universe of recipients to which these reforms apply, and (2) select the reporting options for the simulation.

The PC-based QC Minimodel is routinely used for producing quick-response simulations for which a sample of program participants is an appropriate database. From this experience, we have learned several lessons that are relevant to the design for the PC implementation of MATH:

- A PC-based model can be much more accessible than a mainframe-based model, and analysts can use well-designed menus to set up and run many quick-response applications without assistance from a programmer.

- A PC-based model can execute quickly and provide rapid policy simulation response.

- Model design and the choice of software for implementing the model are critical to time and cost savings. Converting the processing portion of the model from SAS to FORTRAN reduced model execution time from hours to minutes.[4]

2. FOSTERS

The FOSTERS model was developed to take advantage of the rich data on FSP eligibility available from the Survey of Income and Program Participation (SIPP) starting with the 1984 panel. FOSTERS has been used to develop the eligibility estimates for 1984, 1985, 1988, and 1989 as reported in the series *Current Perspectives on Food Stamp Program Participation* (FNS, 1988, 1990, 1992a and 1992b). The model replicates as closely as possible the eligibility determination process of the program. It also models the unit's participation decision based on reported participation status and the observed probability for units with like characteristics. The model was originally developed for the mainframe computer but was moved to the PC in 1989. Both versions of the model were programmed in FORTRAN.

Simulations are normally based on a full sample of SIPP households reorganized in a household-level format. A simulation for 10 plans executes in less than 15 minutes on a 486 machine, which would cost under $4 at MPR's rates. The mainframe version executes in about the same clock time, but would cost $25 for a single reform plan simulation. FOSTERS is neither parameterized nor does it have a developed user front end for setting up simulation runs. By parameters, we mean the variables that define an assistance program. For instance, some of the variables that define the FSP are income limits, assets limits, and benefit levels. If a model is "parameterized," we can change the values of these variables (or parameters) without changing computer code. The following lessons have emerged from the development and operation of the FOSTERS model:

[4]The SAS version would need 1 hour to run one reform plan on a 386, while the FORTRAN version would take under 10 minutes to run 5 reform plans on the same computer. The SAS version could only run 1 plan at a time, so 5 reform plans using the SAS model would take about 5 hours.

- Complex eligibility simulations using the rich SIPP data and associated long records are feasible on a PC.
- Hardware and software are readily available on the PC for this type of application.
- Significant time and cost savings can be easily achieved by moving a simulation model from the mainframe to the PC.

3. SPSD/M

The Social Policy Simulation Database/Model (SPSD/M) was developed by Statistics Canada (1990). It is implemented on an IBM-compatible PC, written in the C language, publicly distributed, and designed to be easily used by nonprogrammers. It is a static model of the Canadian household tax and transfer programs. Several lessons from the SPSD/M model are relevant to this design effort:

- Relatively complex program simulations can be set up interactively. The SPSD/M system allows users to interactively set up the model parameters for the simulation.
- Interface with other MS-DOS software packages such as PC SAS and Lotus 1-2-3 is feasible and attractive to users.
- Models implemented on the PC that emphasize user access are used widely. Statistics Canada reports that there are numerous users of the SPSD/M model outside the agency.
- PC-based models of several complex tax and transfer programs can provide quick response. Statistics Canada reports that a typical simulation requires 20 minutes on a 386/20 machine.

4. TRIM2

The Urban Institute has ported part of the mainframe and VAX version of their simulation model, TRIM2, to the PC.[5] The Supervisor (the framework for the model) and its attendant components have been converted to the PC, and an employer health insurance module is working on a PC. To achieve this change, The Urban Institute had to convert the IBM assembler language routines, which are specific to the IBM 370 mainframe series, to a more portable language. The

[5]To port, or to port code, means to take code from one platform and make it operate on a different platform.

Institute also was required to rewrite the data management facilities built into the mainframe version. The C language was selected for both tasks. The model runs in a very impressive 30 minutes on a 25MHz 386 PC using a 30MB version of the database. This version contains the full CPS household sample, but only the most commonly used variables. These results are important in terms of assessing the feasibility of a PC-based MATH application, since the TRIM2 model is very close to the MATH model in concept and magnitude. The TRIM2 model demonstrates that:

- A PC-based application of a model of the size and scope of the MATH model is feasible.
- It is possible to convert the IBM mainframe assembly language routines.

5. CORSIM

CORSIM 2.0, developed by Steven B. Caldwell at Cornell University, is a dynamic microsimulation model of the U.S. population. CORSIM is written in the C language and is portable across a range of computational platforms. It models fertility, mortality, immigration, first marriage, assortative mating, divorce, remarriage, and home ownership, as well as many other dynamic processes. CORSIM was coded, debugged, and calibrated on desktop PCs. It can generate 30-year simulations with an initial sample of 50,000 or so persons on a 25MHz 486. For larger runs, the model is ported to other platforms, specifically Cornell's IBM 3090/600, since these runs exceed the capacity of PCs. The lessons generated by the CORSIM model are:

- There are some limits to modeling on a PC, although it can handle very large simulations.
- Code can be written so that it is portable.
- Large models can be developed on PCs and uploaded to a mainframe and vice versa.

III. THE CURRENT MATH MODEL

The current MATH model is a comprehensive simulation system designed to estimate the distributional impacts of reforms to major transfer programs (Doyle et al., 1990). The system has two fundamental components: (1) the microlevel database derived from a nationally representative survey of households and persons and (2) the software designed to manipulate the data, to replicate tax and transfer program eligibility criteria, and to simulate behavior. Although numerous functions have been appended to the system over the years, the main purpose of the system is to simulate federal income and payroll taxes, major cash welfare programs, and the FSP. In this chapter, we describe the microlevel database; the primary functions of the model; its development, design, and operating environment; and its strengths and weaknesses.

A. MICROLEVEL DATABASE: THE MARCH CURRENT POPULATION SURVEY (CPS)

The microlevel database is typically derived from the March Current Population Survey.[1] This national survey of approximately 55,000 households collects information on the demographic characteristics and labor-force participation patterns of the population as of March of each year. It also collects information on income and labor-force participation for the preceding calendar year. This survey has been the primary database for microsimulation modeling since the early 1970s, and its large sample size ensures that it will continue as such in the immediate future.

B. FUNCTIONS

Although the March CPS has much to offer, it has a number of weaknesses in serving as the basis for simulating tax and transfer programs. As a result, a substantial portion of the model itself is devoted to compensating for those weaknesses through data preparation functions (function 1).

[1]Other sources have been the Survey of Economic Opportunity (SEO) and the Survey of Income and Education (SIE).

Once the weaknesses in the database are addressed, the model simulates tax and transfer programs (function 2) under current regulations in order to establish base-law programs. Because the software used to simulate base-law programs is parameterized, program values such as asset limits can be easily modified so that the model can better simulate such reforms as eligibility and benefit determination (function 3). Finally, the software and databases are archived and maintained so that the results of earlier policy simulations can be reconstructed (function 4). The model functions, pictorially represented in Figure III.1, are discussed below.

1. Data Preparation

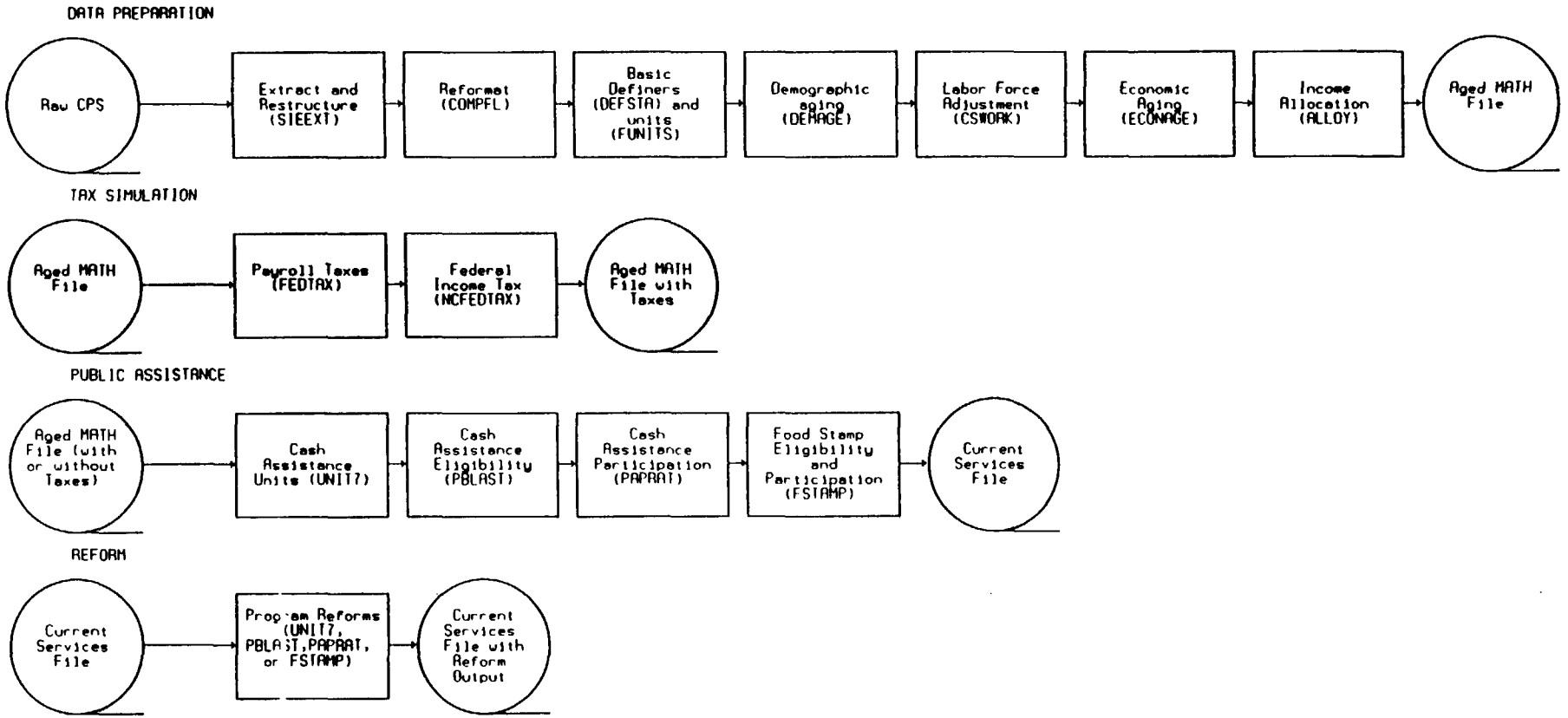
Data preparation can be classified into five subfunctions: (1) converting the raw data to the format underlying the MATH system, (2) reorganizing the information to conform to the input requirements of the MATH model (also known as recoding), (3) aging the data to a time period representative of the circumstances when proposed reforms are likely to be implemented, (4) allocating annual income to monthly amounts, and (5) imputing missing information needed to determine program eligibility and benefits. With a few exceptions, these data preparation functions are carried out in separate modules within the system and are executed before the simulation of taxes and transfer payments.

a. Conversion

The MATH model has internal data management functions that require the data to be stored in a special internal format. The special format serves two functions: efficient internal computations and data compression. Numbers are stored as either four-byte floating point (real) numbers or as one-byte binary integer numbers. Binary numbers require no translation to be usable by FORTRAN and hence are more efficient. At the same time, they use less storage space than character data. For example, a three-digit number (up to 255) can be stored in one byte in binary.

FIGURE III.1

FUNCTIONS OF THE CURRENT MATH MODEL



An important aspect of the MATH file format is that it is self-contained. That is, the file contains a built-in data dictionary, which includes the following description information about the file: the names of variables on the file, the type of each variable, the number of variables on the file for each record type, text records describing the file, file origins, and the simulation year.

b. Recoding

The model performs two types of data recoding: data restructuring and variable creation. In the data restructuring:

- The raw CPS household-family-person hierarchy is converted to the MATH family-person hierarchy.
- Records for children younger than 15 are "shortened;" that is, the model deletes blank variables that do not pertain to children (such as income, labor force, and disability data).
- Unrelated individuals younger than 15 are assigned to adults or families in the same dwelling unit to form pseudofamilies.
- Large households are split because in this case, the arrays that hold person and family data within the model were limited to reduce memory use.[2]

As it creates variables, the model creates basic definer variables and identifies persons to be included in the nonpublic assistance transfer units.

c. Aging

Since the government sets policy to be implemented in the future, it is important to project the impacts of proposed policy changes forward in time. The MATH model does so by aging the data before simulating base-law and reform programs. The MATH model uses a comparative static approach in aging the data. That is, the sample is reweighted to reflect the expected change in the size and demographic characteristics of the population, and characteristics of individual observations

[2]The mainframe model is limited to 7 families and 20 persons. We plan to remove these limits in the PC model.

are then altered to achieve projected aggregate unemployment rates and incomes by source. The target year to which the data are aged is referred to as the simulation year.

d. Income Allocation

The CPS collects data on annual income for each person age 15 or older. In contrast, the need-tested programs simulated by MATH use a monthly accounting period in measuring income for each program unit. The MATH model imputes an intra-year income stream to each person based on observed intra-year income flows in SIPP (Doyle and Trippe, 1991). Person-level monthly income is then aggregated to the unit level to determine benefits.

e. Imputation

The CPS omits some key determinants of program eligibility, benefit levels, and tax liability. Hence, the model includes procedures to impute the missing information. These procedures vary in their complexity and objectives as illustrated below:

- The model imputes financial and vehicular assets to families based on a six-equation system developed from 1985 SIPP data.
- The model imputes child-care expenses to families based on a two-equation system developed from 1985 SIPP data.
- The model imputes medical expenses to households based on the 1980-81 Consumer Expenditure Quarterly Interview Survey.
- The model imputes shelter costs to households based on equations developed from the 1983 American Housing Survey.
- The model imputes capital gains to tax units as a function of adjusted gross income based on a sample of individual income tax returns published in the *Statistics of Income*.
- The model imputes the decision to itemize deductions and the level of itemized deductions as a function of adjusted gross income based on a sample of individual income tax returns published in the *Statistics of Income*.

2. Tax and Public Assistance Simulations

The core of the MATH model contains modules that simulate tax liabilities, cash assistance programs, and the FSP. In each instance, the model processes persons grouped into program units, determines eligibility, calculates benefits, and determines participation. These variables are appended to the individual records in the database.

a. Tax Liabilities

MATH simulates federal individual income tax liabilities and payroll tax liabilities. The procedures for the former are first to organize members of dwelling units into tax filing units based on marital status, whether they are head of the household, and dependency within the household. These units are then subjected to the tax formulas expected to be implemented for the simulation year. Tax liabilities and credits are then created, appended to the microdata file, and tabulated across the sample. User parameters govern tax rates, deductions, credits, and taxable income.

The simulation of payroll tax liabilities encompasses the Social Security, Railroad Retirement, and federal employee retirement program payroll taxes. The simulation replicates program rules for each individual wage earner, with parameters governing tax rates and taxable income.

b. Cash Assistance Programs

MATH simulates eligibility, benefits, and participation under the Supplemental Security Income, Aid to Families with Dependent Children, and General Assistance programs. The model first organizes persons in Census dwelling units into program units based on family relationships, disability, labor force activity, and dependency status. The model then computes program eligibility based on the income, deductions, and asset holdings of members of the program units and of persons whose income is allocated to program units by virtue of their relationship to its members. Participants are then randomly selected from among eligible units with probabilities based on eligibility group within the program, reporter status (that is, whether or not they reported welfare participation), location,

and benefit levels. Parameters govern countable income, income limits, countable assets, asset limits, benefit reduction rates, deductible expenses, other deductions, income exclusions, credits, benefit levels, and maximum benefits.

The cash assistance simulation modules were just recently revised to alter the way in which the simulation deals with the accounting period issue. The new module operates on monthly income and produces monthly benefits. (The old module operated on annual income converted to weekly amounts and produced annual benefits.)

c. Food Stamps

The MATH model simulates eligibility, benefits, and participation under the FSP, as well as benefits and participation under cash assistance programs. This function is performed as follows. First, the model forms food stamp units. Second, the model computes eligibility and benefits based on the income, assets, and deductible expenses of the units. Finally, the model randomly selects participants from among eligible units based on income, household size, age, and participation in public assistance programs. The user can control most aspects of the simulation through parameters that govern income and asset limits, unit formation, deductible expenses, and other eligibility provisions.

3. Reform Simulations

The essence of the MATH model is to produce a before-and-after picture of a program reform. By comparing the two pictures, we can measure the impact of program reform. To ensure that only the impact of the proposed change is measured (not the effects of inconsistencies or errors in the underlying data), we first simulate base-law program regulations (or regulations expected to be in existence in the simulation year). Each of the tax and transfer program modules is executed in sequence yielding a "current services" file; that is, the raw data are aged to the simulation year and appended with tax liabilities and program benefits determined under the program rules legislated to

be in effect that year. These program simulations typically yield estimates of the current caseload. These estimates vary somewhat from program data, but they are consistent with information in the underlying database. This consistency is an important element in reducing error in the analysis of the impact of program reform.

To model the effects of a program change or set of changes, we execute the model on the current services file, changing the parameters (or sometimes the software) of the eligibility and benefit formulas to reflect the proposed program change. (The only difference between the proposed program and the base-law program is the proposed reform.) Participation decisions under the reform environment are conditional on the original base-law program participation decision, and on changes in eligibility and benefits caused by the reform. The results are appended to the database and tabulated to illustrate both the net impacts of the program reform and who would gain or lose as a result of the reform.

4. Archival and Maintenance

The ability to respond within 24 hours of a request for estimates of the impact of a program reform is essential to the MATH model. This ability is attributed partly to the flexible design and parameterization of the system and partly to a fail-safe system for archiving and maintaining the software and the data set.

The model itself is "frozen" each time we update the database and MATH model. In other words, no changes to the system are made unless absolutely necessary. Therefore, we can compare all estimates of program impacts generated from the same model database to one another and to the base-law program without being concerned about artificial impacts caused by changes in the code or database. Since a new database is built about once every three years, there is a considerable interval between model updates. We continue to enhance the system during this time, but we perform the software changes on a second copy of the model, which is not used for quick response. Both the original model and the copy are kept on disk and backed up on three different cycles. A backup is

performed twice weekly and kept for three weeks. A backup is also performed monthly and kept for two years, and finally, a backup is made yearly and kept indefinitely.

Each current-services file is backed up in a number of ways. First, we create a second copy of the full file. Second, we create a "quick-response" file, containing all observations in the lower half of the income distribution. Third, each time a reform plan is executed, we retain the output file, which consists of all of the data on the current-services file as well as all of the reform plans simulated to date.[3]

C. DEVELOPMENT, DESIGN, AND OPERATING ENVIRONMENT

The origin of the MATH model provides insight into its current operating environment and configuration. In this section, we review the development of MATH and how early designs influenced the current version of MATH (Lewis and Michel, 1990). We then describe the current operating environment and configuration, focusing on the model's functional design and structure, user interface, and data management facilities.

1. Development

The roots of MATH go back to 1969 when the Reforms in Income Maintenance (RIM) model was developed for the President's Commission on Income Maintenance Programs. By 1971, the continuing demands on RIM to simulate welfare reform proposals were outpacing the model's capabilities. It soon became increasingly apparent that either the model should be modified to have a more general framework in order to address new modeling requests more easily, or a new model should be developed. After a failed attempt to modify RIM, The Urban Institute launched an effort

[3]Because of limitations in the size of the MATH file, we periodically start over again with a new file that retains only the base-law program current services variables.

to design a new model with its own funds and with support from the Office of the Assistant Secretary for Planning and Evaluation at the Department of Health and Human Services. The result of this effort was the TRIM model, completed in 1973.

One of the biggest improvements of TRIM was its more flexible, modular design. It allowed the analyst to request multiple reform simulations in one run or to use output from one simulation as input to another. Each module, commonly called a *master routine*, could operate on any TRIM-defined data file, and could work independently or with any other master routine. An analyst or programmer could identify which modules the model should run by listing them in a *run sequence*.

The run sequence contained a list of which master routines to execute. Each master routine contained a list of which parameters could be altered. The extensive use of parameters in the various master routines allowed for easier execution. By changing the parameter value(s), a programmer could specify the values for a program simulation rather than modify the FORTRAN code.

Another big improvement was the creation of a group of programs, collectively called the Supervisor, which provided the general framework for implementing the modular approach. The Supervisor had many functions. One was to interpret the run sequence statements and allocate the resources required by a particular model request. Another was to call each requested master routine in a manner that made the best use of computer resources. The Supervisor approach utilized a computer's resources, specifically computer memory, much more efficiently. Because of computer memory limitations at that time, the modelers focused on staying within the 500K limit imposed by the IBM 360 operating system.

In 1975, a group of analysts and programmers at MPR began to develop the MATH model. Building on the TRIM foundation, they developed new capabilities for aging data, and for modeling the FSP and household energy consumption.

2. Current Operating Environment

The team that initiated the development of the MATH model at MPR comprised much of the design team for TRIM at The Urban Institute. They adhered to the model design and programming conventions established for TRIM. The operating environment and general framework of the early TRIM design provided a framework that could support MATH development for almost two decades. Even today, we continue to use the same conventions and Supervisor routines designed and written 20 years ago. Once FORTRAN 77 compilers were available, modifications to the code were made as time permitted to take advantage of the new language constructs, making it easier for the programmers to design more structured, more maintainable, and more efficient code. However, we must rethink these conventions as we consider the movement of the MATH model from its current operating environment to the PC environment. To show how the MATH model works in its current environment, we discuss its modular design, user interface, and data management.

a. Design/Operation

The MATH model structure is similar to a wheel with many spokes. At the hub of all activity is the Supervisor -- controlling, receiving, verifying, and sending household data and parameters to and from each master routine specified in a run sequence. The household data, consisting of family- and person-based records, are passed from one master routine to the next, thereby tying all the pieces together.

The process of running the MATH model involves setting up the run sequence, submitting the job to be executed, and reviewing the model results. These aspects essentially form the user interface. The run sequence describes which master routines to run and the order in which they should be run. The parameters associated with each master routine modify default model values (like the earnings deduction rate, the asset limits, maximum benefits for each HH size, etc.). The job control language (JCL) identifies which file to use as input and which file to use as output. Once the run sequence is established, the user submits the model on an IBM MVS operating system to the

batch queue. We usually run our test jobs on a subset of the file during the day and submit our production jobs on the entire file at night to take advantage of the discounted price during off hours.

A MATH model run has three phases. In the first phase, the model verifies user input, initializes data, and verifies that all variables required for a given module are either present on the database or can be created during processing. In the second phase, the requested simulations for each household are performed. Each routine subjects the household to various tests (for example, to determine if the household is eligible for a transfer payment) and computations. At this time, the output variables are calculated and written to the database. In the final phase, the important information concerning that master routine is summarized, and statistics of interest are printed.

The printed output provides an audit trail that identifies the master routines that were invoked and the parameter values for these routines. For special households chosen by the user, all family and person variables are displayed as are the results of intermediate computations (debug prints). These debug prints are one way in which the analyst and programmer can verify that the model was set up and executed as intended. The summary statistics are another way to do this. In addition to showing aggregate counts, other output may show how many families or persons were affected by the reform.

b. User Interface

The MATH user interface contains two elements: job control language (JCL) and the run sequence statements. JCL, which is required to run programs in batch mode on an IBM mainframe, is the user's link to the mainframe operating system. Through JCL, the user chooses which file will be input and which will be output. The run sequence statements, on the other hand, give the user control over the internal processing by the MATH model. As shown in Figure III.2, the run sequence identifies which master routines to execute and the values of all parameters.

The displayed run sequence indicates that two master routines are to be called by the SUPERVISOR: FSTAMP and WRFILE. RDFILE (the module that reads the database) is always

FIGURE III.2

MATH RUN SEQUENCE EXAMPLE USING RDFILE, FSTAMP, AND WRFILE

```
TITLE    REFORM PLAN 63
CHECKPR                1
SIMYR               1991
DEMAGE                 0
ECONAGE                0
ABORT                  9
PRFREQ            001000
ALTSEQ                 3
RDFILE   FSTAMP   WRFILE
MRVALUES RDFILE          7
FILESRCE               9
FILEAGYR            1991
PRLEVEL                1
PCTPROC              1.0
NPROC             999999
NSKIP                  0
FILEUNMP             5.0
MRVALUES FSTAMP  1      07
PLANNAME
'PLAN 063  DECREASE THE MAXIMUN ALLOTMENT (COUPVAL) '
PRLEVEL                9
ELIGOPT                5
BASELAW        FSBON001
PLANNAME       FSBON063
FSPARTSV       FSBON063
COUPVAL               27
    101.    129.    154.    188.    239.    285.    268.    340.
    407.    342.    434.    519.    404.    514.    615.    486.
    617.    738.    537.    682.    815.    615.    781.    934.
     77.     98.    118.
ENDPARMS
```

not included in the list of modules to be run. The parameter NPROC identifies how many records should be processed. FSTAMP is the food stamp module, and WRFILE indicates that the user wants to write an output file. The parameter COUPVAL identifies which food stamp coupon values should be assigned. These master routines and their parameters are described in the *MATH User's Guide*.

c. Data Management Facilities

Data management facilities is a broad term with several meanings. We define the data management facilities of the MATH model in terms of how the model database and runs are read, written, stored, and maintained.

The MATH model database is a collection of CPS household survey variables (data elements) and simulated variables. The simulated variables are produced by the various MATH routines. In a typical model run, all of the input variables are written as part of the output file. Therefore, the output of one simulation run is the input of another. By reading one run's output as input, the model can more easily compare outcomes from one run to the next and allow income streams generated as output from one run to be input to the next run. For example, simulated AFDC receipt from one run can be income in a food stamp simulation in the next run.

The Supervisor uses the RDFILE routine to read this special file. The RDFILE routine uses two road maps to determine the location of the fixed and floating variables. The MATH model concordance, which is a standalone file, identifies the fixed variables, while header records on the MATH database identify the floating variables. Any interaction with the MATH model database requires the use of RDFILE. For this reason, no off-the-shelf packages can directly read the MATH database.

The MATH model database is written using a special routine, WRFILE. This routine stores the data as variable-length family and person records in a special format to conserve space. Many variables on the file are stored in fixed locations to enhance computational efficiency. The remaining variables are allowed to "float." For example, the variable FSBON001, which represents base-law

Table of Contents

1. Strengths

- **The modular design** makes the model flexible in its ability to respond to policy questions. The fact that modules may be run independently of one another has made it possible to modify or add new routines when policy needs change. Modular design also enables the model to execute several routines in the same job, which reduces execution costs and programmer time.

- **The high degree of parameterization** allows analysts and programmers to quickly implement many of the requests we receive for reform plans. Parameters minimize the chance for errors by eliminating the need for programming changes. Parameters also make it easier to provide better documentation and audit trails.

- **Internal data management routines** automatically manage adding new variables and writing them to a file. The read routines are able to read files created in previous runs, to identify what variables are on the file, and to ensure that variables can be located by all the routines that need them.

- **Use of the mainframe's Tape Management System (TMS)** for archival of MATH databases and file backups greatly eases data management chores. Tape storage in an air-conditioned and secure environment minimizes data loss. In fact, no file or program has ever been lost.

2. Weaknesses

- **Much of the code is 20 years old** and does not take advantage of modern coding techniques. The code has been modified so many times that its logic is difficult to follow and modify. There is a lot of code for policy or program changes that are no longer relevent.

- **Built-in limits on file and array sizes** are left over from the days when memory limits were far less than they are today. The memory limits make multiple full-year simulations almost impossible.

- **The model is expensive to execute** on the mainframe and as a result, does not provide as rapid turnaround as would be possible on a PC platform. Unless there is a dire need to run a simulation during the day, we try to take advantage of night discounts on computer charges. The need to keep costs down reduces our ability to test and perform validity and sensitivity testing. It also extends the working hours of programming staff while they wait for a run to execute in order to check that it is correct.

- **The interface is not user-friendly**. The mainframe environment is not conducive to developing a user-friendly interface. Even experienced MATH programmers could make use of an interface that provides documentation at their fingertips, or cut-and-paste type features for parameters. A lot of time is wasted looking up parameter formats or finding the right input file names.

- **File structure precludes an easy link** to other software packages. While it is possible to do so, the mixture of text and binary data in the current design make it costly and time-consuming to convert the file to a format that can be read outside of the model. This limitation plus the cost of mainframe model runs impedes validity testing and the ability to provide better information on the error associated with model estimates.

- **The use of assembly language** makes it difficult to port the model to other operating systems or platforms. We cannot share the system with anyone unless they have an IBM mainframe.

IV. DESIGN GOALS FOR MATH/PC

While the current mainframe-oriented CPS-based system has served us well for the past two decades, it is now feasible to aim higher in terms of increasing speed and efficiency of the program simulations, enhancing the flexibility of the system, increasing user access to the system, reducing the operating costs, and enhancing the ability to conduct model validation projects. Another important goal is to reduce the turnaround time for responses to requests for reform simulations. We routinely respond within 24 hours to requests for impact estimates of changes to the FSP, but we often cannot respond within 24 hours if the proposal involves multiple passes of the database (typical of reforms involving multiple programs). Model runs (with some exceptions) are executed at night on mainframe computers because they consume large amounts of machine resources and are thus expensive to run.[1]

We believe that microcomputers have the capacity to meet these goals, and we expect to eventually move all of our simulation capabilities to the PC as microcomputers become faster, cheaper, and more flexible. The question is whether now is the right time to begin and if so, what should the initial PC-based model look like and what should its capabilities be? This chapter discusses design goals in two contexts: the eventual full move to the PC and an initial two-platform system.

A. FULL MOVE TO THE PC

Our ultimate goal is to replicate the full power of the MATH model on the PC while greatly improving the model's flexibility, ease of use, capability for expansion, and maintenance. Goals for the PC-based model are a modular design that permits flexibility and timeliness in responding to policy questions; improves ease of use; is easily documented, understood, and accessed; is quickly

[1]Food stamp reforms simulated on the quick-response file are an exception.

updated under severe time pressure; and facilitates access to the data for purposes of validation and analysis.

1. Overall Design

We will group the model's functions into modules with common goals and then order them in a chronological sequence. Each module will be executable as needed, independent of the other modules in the system, and each module will be parameterized to allow flexibility in simulating program reforms.[2]

When fully implemented, the model will contain a supervisory routine to control its components, input/output modules (that is, data management), a user interface designed to facilitate access and use of the model by both programmers and analysts, and a series of routines focused on a specific goal such as simulation of the FSP. They will be ordered in the following manner:

- Data preparation functions (basic recoding and conversion to internal format)
- Aging
- Monthly income allocation
- Imputation for missing assets and deductible expenses
- Program simulations
- General-purpose routine for ad hoc analysis

Each of the modules will be parameterized to the extent feasible. The user interface will allow both analysts and programmers easy access to the parameter descriptions and default values, and it will make it easy to edit the parameter values and select modules to execute. The general-purpose routine will provide an option to generate ad hoc tabulations from the model database or to simulate an assistance or tax program not programmed elsewhere in the model.

[2]Some modules must be run after at least one execution of other modules. For example, the Public Assistance Module (PBLAST) must be run after the Unit Creation Module (UNIT7).

The model should be designed to operate quickly and efficiently. Our goal is to ensure that we can simulate the equivalent of five typical FSP reform plans in less than two hours.

2. Supervisor

The model should have a supervisory routine to coordinate all of the model functions: to call modules, as specified by the user, to be executed in a given model run; to call the data management functions; and to link the user's parameters to the substantive modules. In essence, the Supervisor should be a skeleton or framework from which all other model functions are executed.

3. Data Management

The data management functions are (1) processing the database, (2) managing the database, and (3) supporting the interface between the PC model and the mainframe model, statistical packages, and specialized software.[3] The purpose of these functions is to provide access to the data for the model itself, for model users, and for other software packages; to allow substantive routines to read and write the data; to read and write data from the mainframe model; and to store the microlevel simulation output for future use.[4] The data management facilities should also allow us to jointly process persons and families within a household, to repeatedly examine the characteristics of the persons grouped into different program units, and to easily model joint decisions. However, households can be processed sequentially, since one household's decisions generally do not affect another household's decisions with regard to behavior, such as participation in need-tested programs, or with regard to the household's labor supply.

---

[3]As we discuss in Section B, the plan for the next step in model development is to move to a two-platform system rather than to attempt to move the entire MATH model to the PC. The interface between the two models is therefore needed only as long as we maintain the two-platform system.

[4]Two of our objectives are competing. There is a need to minimize processing time to facilitate quick-response activities, thus the requirement to optimize the data formats and data access procedures. (This optimization typically leads to specialized formats and structures that cannot be interpreted by software other than this model.) On the other hand, there is a need to facilitate model validation and user access to the data. This can most easily be accomplished by formatting the data so that it is easily transported and described.

Our goal in designing the data management functions is to keep the logical structure of the data as a hierarchical file, with households, families, adults (age 15 and older), and children being the four distinct record types. Households are the interviewing unit of the CPS and form a natural unit of analysis for food stamps and for certain behavioral responses, such as participation decisions. Families often represent economic units within households. Persons are the basic building blocks for program unit formation and tax and transfer program simulations. We distinguish between younger and older persons simply because the March CPS collects very little information on children younger than 15; we can therefore use shorter records for children to reduce the file size.

While we want to maintain the logical structure of the data, we may want to employ a physical structure designed to optimize processing speed and minimize storage needs. We also want to be flexible in our ability to read and write data. We need a facility to process CPS-based files that range from a full complement of variables on a full sample of households to a small subset of variables on a small extract of households and persons. The data management functions should also include the ability to append simulation output to the model database and to read it in a later run. There is a need for the ability to write out only the simulation output, to operate on two model databases in tandem, and to read and write data in a format that balances the need for efficient I/O. Also required is the ability to read and write data in a format that statistical packages can read and that users can work with without using the model. One of the goals of our new design will be to balance the need for efficient I/O with the need to easily transport the data.

Because of the sequential nature of the modeling process (households function independently of one another), sequential access techniques are better than random or direct access methods for this application. Since we usually use most of the data for most of the households in a given model application, the model does not constitute an efficient use of random access processing techniques.

4. Statistical Reporting

The statistical reporting functions of the model should include preparation of the standard summary tables similar to those currently produced by the mainframe model (see Appendix B for an illustration of the summary tables produced from the food stamp module). Other required reporting features are the ability to (1) load the standard summary tables into a spreadsheet, (2) print the standard summary tables in a clear, concise format, and (3) generate ad hoc tables and statistics from the reform output and the information on the base-law program database. Ad hoc statistics can be generated in several ways. For instance, a link can be provided between the microdata file output and a statistical package, or between the summary tables and a spreadsheet package. Another way to generate ad hoc statistics as well as ad hoc simulation is to include a general-purpose routine in the overall design.

5. User Interface

The model will be used to support program analysts directly or indirectly in simulating the impact of program reform. Thus, the program analyst must be able to change the specification of program eligibility criteria or the benefit determination formula and to run the model to ascertain the effects of the change. In addition, the analyst needs access to simulation output in various forms both for informational purposes and for further manipulation. Furthermore, the analyst will often need the impact estimates in less than 24 hours, which means that program simulations must be set up and run quickly and efficiently. Finally, the analyst needs an audit trail of the model specification, execution, and output (both statistical summaries and microdata).

Given these requirements, the fully implemented model must have a user interface with two facets: one that supports direct access by persons with little computer experience and another that responds to the needs of experienced programmers. The novice needs a menu-driven system with fully documented options and easy-to-make and correctable choices. The veteran programmer needs quick access to the model and to the parameters. The interface for the novice will therefore be user-

friendly and will make use of screens that present a series of parameters and other choices requiring user input. The screens should:

- Identify the options, current defaults, and prior user choices for multiple program simulations in the same request
- Permit the user to select files or samples for execution (for example, test versus production, or one year versus another)
- Allow the user to execute the program simulation in test or production mode
- Provide access to the statistical reporting functions of the model
- Allow the user to view the summary output on the screen, route it to a printer, or import it to a spreadsheet package
- Perform an archival function by linking the model output to the reform specification and by storing the results
- Allow the user to determine the treatment of the data file (that is, to choose among the output options discussed under data management)

The facet of the user interface geared to the veteran programmer should not only reflect the choices provided by the first facet, but it must also allow the programmer direct access to a database containing the parameter settings so the parameters can be edited quickly. The programmer has no need to pass through the screens with their fuller explanations of options, and, in fact, the time required to do so is a hindrance. The second facet of the interface should also allow the programmer direct access to the software underlying the program simulations so that he or she can make changes to the program that cannot be made through parameter settings alone.

Assuming the statistical reporting goals are achieved in part by creating a link to statistical and spreadsheet packages, the user interface should provide a link to the software for preparing the ad hoc request.

6. Documentation

Our goals for the full system are to provide extensive documentation on all aspects of the system, and to make this documentation accessible, manageable, readable, and easy to maintain. To meet these goals, we will need to maintain the documentation for the file and the model in machine-readable as well as hard-copy form, to develop software to extract pertinent subsets, and to search the document for answers to specific questions. The scope of the documentation should include:

- Model design, function, and uses
- Indices of variables and cross-references on variable usage and origin
- Descriptions of all variables extracted from the CPS or created by the model, including the mnemonic, code number,[5] topic, value ranges, universe, type, and origin
- Parameter definitions, purpose, and defaults
- Substantive algorithms used in the program simulations (program specifications)
- Programming and naming conventions
- Information about software
- Operation of the model and user interfaces
- Audit trail for program simulations, both those executed in the creation of the base law program file and those used in the simulation of program reforms
- Log of model output

7. Archival and Maintenance

We have three objectives in designing the archival and maintenance system. First, any given model run must be replicable at any time in the future. Second, there must be access to output from all model runs for a period of five years after model execution. Third, the system must be continually

[5]Each variable in the system will be identified by a name and by a code number which denotes its origin.

enhanced as new research is completed and ready for implementation. These objectives dictate the following practices, which are repeated for every update cycle:

- At the beginning of the update cycle, create a newly updated model and a base-law program database
- Create four copies of an official model (one for reform simulation, one for model enhancement, and two for backup) and three copies of the database using different storage media, storing one copy of each off-site in a fireproof facility
- Use the first copy of the model as the basis for all program reform simulations during the three-year period
- Create one original and one backup copy of the reform simulation software on two different media, to be stored for five years, one copy on-site and one copy off-site in a fireproof facility
- At the end of the three-year cycle, when it is time to replace the model, create two permanent backup copies of the retiring version of the model on two different media and store one copy off-site
- Continually update the second copy of the model software and parameters as new research on modeling procedures is completed
- At the end of the cycle, replace the retiring model with the copy that has been continually enhanced

B. INITIAL TWO-PLATFORM SYSTEM

We are not proposing to immediately implement the full-scale model on the PC. Instead, we recommend an initial system that satisfies FNS's immediate needs for reform simulation capacity and that permits expansion to the full model at a future time. Below we discuss our design options for the initial two-platform model, the associated model databases, and the design issues surrounding the link between the models operating on two platforms.

1. Optional Views of the MATH Model

In Chapter III, Figure III.1 shows the model subdivided into four functional components. First, data preparation functions are executed once each time we build a new MATH file for FNS (typically every three years). Second, routines simulate tax programs, also executed once each time we build

a database. These routines are used infrequently to simulate program reform or to examine the interaction between tax and transfer programs. Third, programs simulate need-tested programs (SSI, AFDC, GA, and food stamps) that are executed once in the creation of the base-law program file. Fourth, the transfer simulation modules (occasionally tax modules as well) are executed in the simulation of program reform.

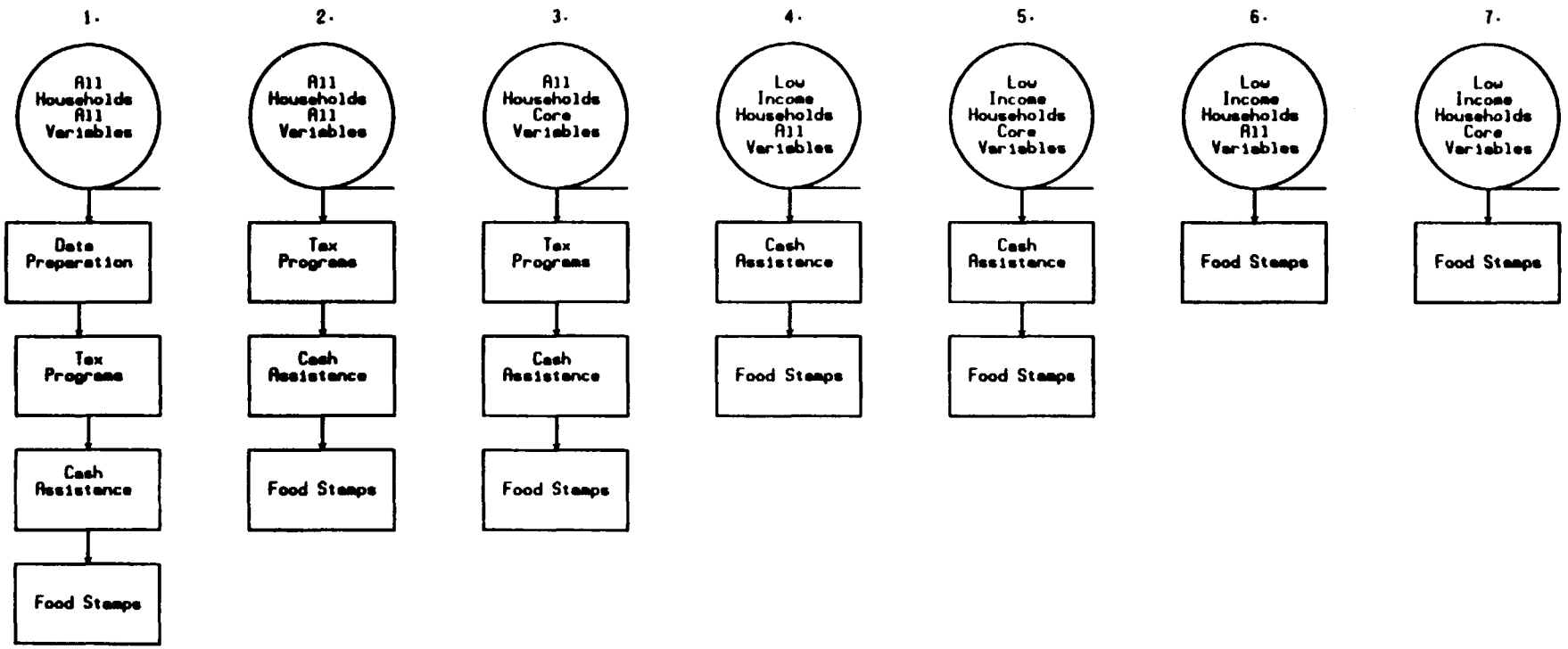
With these functions in mind, we have developed seven options, illustrated in Figure IV.1, for the initial implementation of the model on the PC. That is, as previously noted, we could consider a range of possibilities from a full implementation on the PC of all MATH model features (option 1) to minimum simulation restricted to the FSP on the full file or a subset of the files (options 6 and 7). For example, in the least ambitious initial PC implementation (option 7), the PC model reads a database of low-income households containing only core variables reflecting the determinants of program eligibility and a base-law program simulation of all programs, simulates a program reform under the FSP, and generates statistics describing the gross and net impacts of the program reform.

The food stamp module is obviously the highest priority for the transfer to the PC since it is the most heavily used component of MATH, and hence the minimum capabilities just described would eliminate the need to use the mainframe computer for approximately half of our current quick-response requests. However, we also simulate reforms to cash assistance programs and examine their effects on the FSP. These applications occur frequently enough that we consider it optimal to include the following capabilities in the PC-based model, in addition to the minimum capabilities just described: (1) simulating cash assistance programs with which the FSP interacts, (2) operating on a full MATH database or on a subset of the file to improve response time, and (3) writing out the simulated output variables for future analysis and archiving. This option for the initial PC model is depicted as a combination of options 4 and 5.

Table of Contents

FIGURE IV.1

OPTIONS FOR THE SCOPE OF A PC-BASED MODEL



2. Optional Views of the Data

Figure IV.1 also shows four views of the database. First is the full file with all information retained for all households and persons (this file ranges in size from 70 to 100MB depending on the number of reform plans appended to the file). This appears as the database in options 1 and 2. Next, it shows a file reduced in size by including only the observations in the lower portion of the income distribution (options 4 and 6, which include all units ever likely to be eligible for even the most generous reform plan under any need-tested program). Third, it shows a file reduced in size by eliminating information not used routinely in reform simulations or to describe characteristics of program participants (option 3). Finally, it shows the intersection of those two restricted files, that is, a file of lower-income households consisting of a core set of data needed for reform simulations and description of the program participants (options 5 and 7).

We can choose the last option depicted in Figure IV.1 and meet our minimum modeling requirements except for the ability to simulate interaction with programs such as AFDC. In other words, we can simulate a large number of reforms to the FSP with a small database restricted to potentially eligible households and to variables describing the determinants of program eligibility. Because of its small size, such a database would require the least amount of storage and the least amount of processing time. However, we learned from the QC Minimodel experience that this approach can be limiting. In essence, the scope of the food stamp reforms that can be simulated with this model is limited to changes in the way current determinants of food stamp eligibility are processed. We cannot, for example, conduct (in a PC model operating on the minimal data set) analysis of reforms that involve income not currently counted by the FSP or that involve changes in the definition of students if these additional variables do not reside in the minimal data set.

Given the limitations that the small file places on the reform modeling capabilities, we do not recommend designing the PC-based model to operate solely on a restricted file (either restricted in

terms of variables or restricted in terms of observations). However, we do want the option to process such a limited file for routine quick-response requests in order to minimize turnaround time.

3. Intermediate Design Goals

We recommend an initial implementation of a PC-based model that has the capacity to simulate cash assistance and food stamps on the full March CPS-based file. This model should be flexible in its handling of the database to permit a full range of options for processing the data from the full file (to create a base-law program simulation) to a file reflecting a subset of the variables and a subset of the observations (under quick-turnaround reform applications). The model should yield two types of output: microdata and summary tables. Furthermore, the model should provide options for generating ad hoc tables either through manipulation of the microdata input or output from the model or through manipulation of the summary statistics.

With this plan, we will continue to operate a mainframe-based system in tandem with the PC model to perform the data preparation functions and to simulate taxes. Therefore, we have an intermediate design goal of ensuring that PC- and mainframe-based models can interact effectively and efficiently.

An effective and efficient link between the PC and the mainframe computer can be maintained partly through the chosen design and partly through the capabilities of the read and write routines. In this case, we specifically recommend the following:

- The mainframe-based model should be used to create an intermediate base-law program file that reflects the data preparation steps, the aging procedures, and the tax simulation.
- The mainframe-based model should write the database in character format for porting to the PC.
- The PC-based model should read the mainframe data and convert it to PC-based file format.
- The PC-based model should be used to simulate base-law cash assistance and food stamp programs.

- The PC-based model should be used to simulate reform-law cash assistance and food stamp programs.

A design with these capabilities ensures that all aspects of cash assistance and food stamp modeling occur on the same platform. As a result, there is no concern about consistency in outcomes across machines. This design also eliminates the need to transfer between platforms in the middle of a quick-response request that involves both cash assistance and food stamps. The read and write routines ensure that on the infrequent occasions when we do need to move between the two platforms, we can do so with a minimum of time and effort.

V. FEASIBILITY OF A PC PLATFORM

Before we could make a final decision about the feasibility of a PC version of the MATH model, we needed to carefully examine our computer options and perform enough testing in order to be confident that a MATH/PC model was practical. We began by looking at the two principal features of any computing platform. The first is the underlying hardware and its speed, and the second is the operating system and the software available for that operating system. Both of these components have to be adequate for our purposes before the platform can be considered satisfactory. MPR's conclusion is that (1) it is very feasible to run a MATH model on a PC and (2) the most appropriate platform for implementing the MATH/PC model is the PC using DOS as an operating system. The PC has more than enough computational power, is the least expensive platform, has more relevant software written for it than any other platform, and will therefore more than adequately serve as the host for the MATH model. This is not to say that the PC will always be our platform of choice, or that DOS is the "right" operating system. However, we can say that a DOS-based PC will allow us to get started quickly, keep development costs to a minimum, and be a reasonable platform for years to come.

PC technology has advanced tremendously since its introduction in 1981, as discussed in Chapter II. The size of the market and intense competition between vendors has driven the rapid development of the PC and workstation technology. While it is difficult, if not impossible, to predict what the PC environment will be like in the future, we can say with certainty that the PC and workstation computing environment will feature larger and faster machines for less money.

We believe the best course is to design MATH/PC to make use of widely available, and hence less costly, DOS PCs. Our strategy will be to test performance on such machines and only move to higher-end, more expensive machines and OS's if necessary. We also want to design MATH/PC to be as transportable as possible to future operating systems and hardware so that moving the model

will be as painless as possible. In this chapter, we discuss current PC technology, our test results, and our conclusions about the feasibility of a PC platform for the MATH model in terms of five key areas: operating systems, computational power, software, data management, and maintainability.

A. OPERATING SYSTEM FEATURES

Table V.1

PERFORMANCE COMPARISONS ON SIMILAR HARDWARE

Hardware Platform	1990 Prices	Integer Operations	Floating Point Operations[b]
AST 486/33	$4,490[a]		
DOS		34,192	1.46
OS/2 1.3		14,494	-
SCO UNIX		35,226	1.52
Club American Hawk III			
DOS	$6,495	35,923	1.58
OS/2 1.3		15,625	-
SCO UNIX		28,250	1.59
Compaq Deskpro 486/332			
DOS	$25,270	33,664	1.52
OS/2 1.3		15,385	-
SCO UNIX		36,977	1.54
UNIX Workstations			
IBM Powerstation	$12,995	45,454	8.15
HP 9000-375	$25,745	17,211	0.421
MIP Magnum 3000	$16,990	37,460	5.727

SOURCE: *Personal Workstation*, vol. 3, no. 4, April 1991.

[a]Motherboard only.

[b]Arithmetic involving numbers with decimal points. The MATH model uses a great amount of floating point arithmetic.

there may be other reasons for such a move. Although unlikely in the near future, it is possible that UNIX may be required by government standards, or a future critical application may be available only on UNIX machines. We therefore include a discussion of the UNIX environment in Appendix D.

In sum, since DOS is the least costly option and provides more than adequate performance, we will probably use it for the near future unless a compelling reason for moving to new OS's arises. Our final choice was based primarily on the following reasons:

- Both MPR and FNS have DOS machines
- MPR and FNS do not have UNIX systems. MPR has a minimum number of Windows 3.1 and OS/2 machines
- Software is less expensive on DOS machines because of greater competition and a large market.
- DOS has the smallest machine requirements of the operating systems.
- The other operating systems would require additional training. UNIX training would probably be the most expensive, since it is the most complex.
- There were 9.8 million single user machines sold in 1991. Of them, 8 million were Intel-based PCs, 1.3 million were Macintosh systems, and fewer than .5 million were RISC (UNIX) systems. DOS's market share is much more inviting than the other markets to software and hardware vendors; thus, there are more applications for DOS machines.

B. COMPUTATIONAL AND INPUT/OUTPUT SPEED

The first question in making a decision about the feasibility of a PC-based MATH model is: does the PC have enough computational power to perform the calculations required by the MATH model? In fact, and perhaps surprisingly, it has more than enough (see Table V.2). For example, we executed a series of mathematically intensive jobs containing far more calculations than the most complex MATH model run on both an IBM mainframe 3081 and on a variety of PCs. The mainframe execution took *twice* as long as our fastest 486. In fact, we had to scale down the test because the mainframe charges would have been in the $1,000 range for something that took less than 15 minutes on a 486 PC. The PC has more than enough arithmetic power.

The second question is: can the PC manage the files the MATH/PC model reads and writes? To answer the question, we created a 147MB test file; then, in a second program, we read that file back in and wrote out a second 147MB file. These files are at least twice as large as a typical full

TABLE V.2

COMPUTATIONAL BENCHMARKS FOR THE SAME FORTRAN MATHEMATICALLY INTENSIVE CODE

Platform	Time (in minutes)	Approximate Cost (in dollars)
Personal Computers		
386 25MHz	3:48	<$1.00[a]
386 33MHz	2:45	<$1.00[a]
486 33MHz	1:06	<$1.00[a]
486 50MHz	0:41	<$1.00[a]
Mainframe Computer		
3081 IBM	1:23	>$80.00[b]

[a]Using MPR's PC cost structure.

[b]Cost at MPR's mainframe service bureau.

CPS MATH database. While the PC demonstrated that it could handle the MATH model file sizes, both the IBM mainframe and the UNIX system we tested were faster at writing data than our fastest PC (see Table V.3). From Table V.3, we see that the I/O times vary considerably depending on the PC compiler and the type of file format used. However, note that the elapsed time for the I/O test on PCs was often about the same as the elapsed time on the IBM mainframe, since the mainframe requires an operator to mount input and output tapes and that many jobs may be executing on the mainframe at the same time.

TABLE V.3

TIME REQUIRED TO WRITE A 147MB FILE USING DIFFERENT HARDWARE PLATFORMS AND FILE FORMATS
(in minutes)

Platform	Format1[a]	Microsoft[b]	Format2[c]
386 IDE[d] 33 MHz	54:18	8:30	8:26
486 IDE 33 MHz	63:42	9:12	11:26
486 IDE 50 MHz	50:39	5:40	7:38
486 IDE 33 MHz with OS/2			
DR DOS[e]	6:30	5:16	4:24
DOS under OS/2	4:32	3:42	4:44
486 EISA 33 MHz	6:49	4:08	3:10
IBM 3081	0:22	--	--
RS-6000 UNIX Workstation	4:35	--	--

[a]A Lahey FORTRAN file format that has no special record markers and can be read by many different packages and languages without any special coding.

[b]Microsoft FORTRAN's version of Format 1.

[c]Lahey's internal format which we believe can be relatively transportable and is fairly efficient. This format has special codes embedded in the data.

[d]Integrated Drive Electronics, an increasing popular and inexpensive drive that includes most of the electronics on the drive itself rather than on a separate piece of hardware.

[e]DR DOS was using OS/2's disk caching which has a significant benefit.

The slower I/O on the PC is a symptom of the rather inefficient DOS file-handling system and some inefficiency in the hardware. Nonetheless, we would expect that a well-equipped PC would be able to read and write a typical MATH database (60,000 households and 70MB of data) in under thirty minutes. Table V.4 displays the results of our most demanding potential MATH/PC applications in column four (worst-case testing). First, we were reading and writing at least twice as much data as we expect to process. Second, when we performed mathematical calculations, it was at least three or four times what we think a MATH/PC run will typically perform. We performed these very demanding tests to be absolutely sure that PCs will be a viable platform. Based on our results, the PC's performance is more than adequate.

TABLE V.4

READ/WRITE PROGRAM BENCHMARKS FOR PCs
(in minutes to process a 147MB file)

Platform	Read Only, No Math	Read/Write, No Math	Read Only, Intensive Math	Read/Write, Intensive Math
386 ESDI 33 MHz	11:52	24:40	49:42	61:46
486 IDE 33 MHz	6:30	30:24	24:00	52:12
486 IDE 50 MHz	6:22	13:06	13:10	18:34
486 EISA 33 MHz	3:36	9:24	14:52	20:44

C. SOFTWARE TOOLS

We intend to make use of off-the-shelf packages as much as possible. As long as a package or tool meets our processing goals and performs efficiently, there is very little reason for us to write our own code.

1. Modeling

MPR has developed several microcomputer models using current PC technology. The MATH model is at least an order of magnitude larger than either of the two other FNS microsimulation models in terms of the total amount of code involved and in the complexity of its design. The size and complexity of the MATH model demand that we be cautious about the phasing of the work. Fortunately, we are able to implement a PC-based model in such a way as to minimize the risk of moving to PC technology. We will do this by (1) using a language we know well, (2) using proven technology, (3) writing transportable code, (4) staging development, and (5) not doing more than is feasible.

Our first choice for the programming language is FORTRAN, which is the language in which the bulk of the MATH model is written. This will facilitate our move to the PC platform and will minimize our conversion costs, since we will not need to completely rewrite the existing code. FORTRAN is oriented toward scientific applications where there are many equations and matrix manipulations of the type we use in the MATH model. Furthermore, FORTRAN is the language used most often by the "Microsim" community and that with which it feels most comfortable, and it is portable across platforms.

MPR has been very successful with previous microsimulation efforts using FORTRAN on microcomputers. Both the FOSTERS model and the QC Minimodel (once converted to FORTRAN) have provided MPR and FNS with very quick turnaround and have been relatively easy to modify. FORTRAN has also been a fairly easy language to teach to junior programmers. The time required for them to become comfortable with the language is minimal. In addition, many of our research staff can read (and write) FORTRAN code, and they are therefore able to help debug programs or find misinterpretations of the program specifications.

FORTRAN has subscript checking (array bounds), which helps programmers tremendously in testing and debugging their code. This is important because an out-of-range subscript can be an

elusive bug, an obstacle to quick-response applications. While bugs are inevitable, it is important to minimize them and to make them easy to find.

MPR has found that FORTRAN compilers are very portable, and with careful attention to coding in a "standard" way, porting to a UNIX, WINDOWS, OS/2, or even a mainframe environment will require minimal effort. If these standard coding rules are followed, we should be able to migrate to any platform that might be chosen at a later date.

C would be the natural alternative to FORTRAN and has proven to be faster in our tests for both I/O and computations. Our tests demonstrated that C is computationally faster than FORTRAN by 10 to 25 percent. However, this edge in compiler speed is overshadowed by the continuing increases in machine speed. For example, if PCs double in execution speed every year or so (which they historically have), there is no need for concern about a 25 percent difference in speed between one language and another.

As one can see from Table V.3, the FORTRAN version of the I/O test took from 3:10 minutes to over 50 minutes depending on the version of FORTRAN I/O used and the platform on which the tests were run. We did not do as much testing for the C I/O benchmark, and we do not profess to having written the most efficient C code, but the I/O time between the fastest FORTRAN and the fastest C tests was less than two minutes--a 33 percent difference. The tradeoff between the two languages is processing speed versus clarity and ease of learning. C has the ability to perform low-level operations directly on the hardware. Microsimulation applications do not need or use this power. C's additional power comes at the cost of more complex commands, more tasks for the programmer, and increased learning time. If the execution time savings were measured in hours or half-hours rather than minutes, C would certainly be a serious contender for the majority of the modeling work. As noted later, we are considering the possibility of using C to replace the IBM assembly language routines, but we are not sure if that is necessary since FORTRAN can now perform many of the functions performed by the assembly language routines.

2. Input/Output

We have spent a little time investigating the computational and I/O speed of the Lahey FORTRAN compiler versus two other FORTRAN compilers: Microsoft 5.1 and WATCOM v9.0. We liked the feel and error-handling capability of the Lahey version better than the other two. Lahey has better computation speed, but lagged in I/O speed compared to the other two compilers. Microsoft's version is not the 32-bit, extended-DOS version that we need for the MATH model, so we tested it only for I/O. The WATCOM compiler was fast, but more difficult to use, and its error messages are not as helpful as Lahey's messages.

We used several different PC platforms for the testing, which are typical of the programmer platforms MPR uses with two exceptions: (1) the hard disks were usually larger than normal, and (2) the single Extended Industry Standard Architecture (EISA) PC we have is specifically designed to speed up I/O.

Our read and write tests on the EISA machine showed an improvement of up to 500 percent over the performance of the fast 500MB IDE drive using the same benchmark program. As shown previously in Table V.4, the EISA machine read and wrote the 147MB file in less than 15 minutes in our most demanding version of the FORTRAN I/O benchmark program. It did the same in under 10 minutes in our least demanding version.

We wanted to create as realistic but as conservative a test as we could that would reflect the execution time of a typical MATH model simulation. We therefore included computations much more intensive than any we have seen in a model run, along with reading and writing our worst-case files. The run times on the various 486 PCs varied from 13 minutes to over an hour. These run times varied considerably depending on such factors as disk free space, read/write techniques, and the number of calculations performed. MPR's expectation is that typical MATH run times on current high-end PCs will be less than 30 minutes. If an output MATH file is not created, the processing time will be reduced to under 15 minutes.

One of the more significant tests that MPR conducted was a highly computational intensive test that ran for about one minute on the 486 PC and for about one and a half minutes on an IBM 3081 mainframe (see Table V.1). The PC run would cost around $1 at MPR's current rates, while the approximate cost of the mainframe job was $80. The mainframe did a far better job on I/O-intensive tasks than on computational tasks, but it is still much less costly to run I/O-intensive jobs on the PC than on the mainframe.

3. User Interface

The user interface should help the user choose which master routines to run, which parameters to alter, what the format for the parameters will be, and which files to use. A menu-driven interface that is either character-based or graphical in nature would be sufficient. Setting up a routine model run should be a simple matter of highlighting or selecting which master routine to use, selecting which parameters to modify, and pressing a help key to obtain additional information about a selected item. The user interface must be faster than the QC Minimodel interface, which is written in SAS.[1] A menuing system written in a procedural language such as C or Visual Basic would be much more efficient than the QC Minimodel process. The MATH model's user interface needs to handle much more information. Indeed, we will want to make the model interfaces across MATH/PC, FOSTERS, and the QC Minimodel have the same look and feel to the extent that resources permit this to be so.

We could develop an interface using any number of tools. Conventional programming languages like FORTRAN or C could be used. Both languages are well supported in the DOS, Windows, and OS/2 environments. Another alternative is one of the dBASE-compatible packages. Since these packages already have built-in screen and data management facilities, the interface development time would be much less than one developed using a procedural language. We feel the best alternative

[1]The QC Minimodel interface consists of a series of screens containing a list of questions. Based on how the question is answered, additional screens display the default parameter settings and allow modification of the settings. This interface takes about seven minutes to complete.

would be to use a language such as Visual Basic. This tool has proven to be a good prototyping tool as well as a developmental tool. Machrone and Miller (1991) report that programming with Visual Basic "... makes programming in Windows easy." They go on to say that the program provides a degree of interactivity that is usually denied to the developer and advanced user. Microsoft is developing a DOS version of Visual Basic in addition to the Windows version reviewed above. We feel a user-friendly interface will be a tremendous benefit to the microsimulation community and that it is only possible on the PC.

4. Report Writing

The current MATH model offers three different ways to report on the outcome of a reform simulation. The first way simply displays summary statistics and gainer/loser tables (for food stamp simulations) as part of the standard model printout. This printout or hard copy can be used directly by the analyst. However, the tables can also be saved in machine-readable form downloaded to a PC, manipulated by an editor, and incorporated into a word processing document or a spreadsheet for additional reporting. The gainer/loser file, which is normally produced as part of the output of the food stamp master routine, is written both in hard copy form or in an electronic form that is directly importable into a spreadsheet package. The second way is to have a programmer use the TALLY master routine to produce ad hoc tables directly off the MATH file. This method provides access to and reporting directly from the MATH files. The final way is to use one of the special options of the WRFILE master routine. This option tells the supervisor to write the MATH file as a standard text file.

Naturally, the MATH/PC model should continue to produce the standard reports as part of its default output, using the language chosen for the modeling functions. These tables would be output in machine-readable form for viewing, printing, and further manipulation outside of the model, all without the extra step involved in downloading. We also support continued use of TALLY to produce ad hoc tables. In addition, we expect to be able to create a file structure that is directly

accessible by statistical packages such as SAS and table-processing language (TPL). This will be a giant stride in making the MATH model's data available for further analytical and validation purposes.

5. Backup/Archival

After using the command that deletes files, the user often has a hollow feeling that might be best expressed as: "Are you sure you didn't need that file?" One only has to lose a file once to know that backup/archival functions are critical. As Mendelson (1991) notes:

> A hard disk failure, a mistaken delete command, a distracted worker who overwrites a long file with a short one that has the same name -- all these common occurrences can make you lose a file as thoroughly as theft, flood or fire.

Many computer users decide to forego their backups and risk a catastrophe because they claim that it takes too long to back up their files, and it is too cumbersome to restore them. DOS built-in utilities are too cumbersome to be practical with the amount of data and files involved in the MATH system. We need to know that the MATH/PC model can be backed up and moved to other systems and that there will be tools available to help us do that. We need the capability to back up at least 300MB of data (a full input and output MATH file) for the MATH/PC model.

MPR currently uses FileSafe software by Mountain to back up the data on our EISA machine (which has a 1GB hard disk) to a DAT drive system with a 2.2GB capacity. The tape drive and FileSafe is capable of backing up 10MB of data per minute (with data verification on, about 5MB/minute). A 300MB subdirectory could be backed up and verified in about an hour.

The second backup/archival medium is the Bernoulli drive with its removable cartridge system. The upper limit of Bernoulli drives is currently 90MB without data compression and 180MB with compression. This new size limit makes Bernoullis more attractive for the MATH/PC model, but again the cost per megabyte is an issue. See Table II.5 for a comparison. Also, an entire application or simulation may not fit on a Bernoulli cartridge, so this option, while possible, may not be practical.

A third type of storage medium is gaining acceptance in the computer industry: magneto-optical (MO) disk drives. These disks go one step farther than the read-only compact disk devices (CD-ROMS) popular for distributing large databases. They allow read and write capability similar to that of a hard disk drive. The drawbacks with this technology are that it is still considerably slower than hard disks, and the media costs are significantly higher per megabyte. However, optical disks are much easier to use than tape since one can copy or write to them, do directory searches on them, and otherwise treat them just like a hard disk. MPR expects to purchase one soon and will then test its functionality for the MATH/PC model.

MPR will have all these options available by the time the MATH/PC model is running, so we can afford to take a wait-and-see attitude for the present. Any one of the alternatives would work for us, but which one is best remains to be seen.

D. DATA MANAGEMENT

There are several aspects of data management that need to be addressed. We will focus on how to store and access the actual microdata file, since that is the only one that would be affected by the platform we intend to use.

Most of the MATH model simulations use the quick-response subset of the database, or about 30,000 households. Every household in the set is processed by the model. This has important implications for the efficient storage of the data. Several of MPR's other projects have involved using and comparing sequential and random-access methods and the relative efficiencies in CPU times and storage costs. It was quite clear from these studies of applications such as the MATH model, where every record needs to be read, that sequential access is much more efficient.

We have seen from the tests we have conducted for this report that I/O costs are a substantial part of the total cost of a MATH run. We also know that smaller files with more complicated file structures are more efficient for I/O than larger files with simpler file structures. Therefore, we are convinced that it would pay a dividend to reduce the file size. However, we also know that the

compact file structure cannot be read outside of the system that created it. Since we believe that storage will continue to become less costly, and processing power will continue to grow rapidly, we believe that in the long run, a simple file structure that can be read by outside applications will be more in the spirit of the Academy's report and will ease computer tasks for the majority of users.

We should be prepared for all eventualities, and we should thus think about the several ways to reduce MATH/PC file sizes. The first method we are considering is to write only nonzero real amounts (and an identifier field) to the file. In a typical MATH file, we calculate that this would save up to 40 percent of the space required to store a full file. A second method would be to store a variable in the least number of bits required to store the variable's range. Unfortunately, because the minimum size of an intrinsic FORTRAN variable is 1 byte, there would be some extra work involved in decoding bits into normal FORTRAN variables. The tradeoff between more complex code and storage efficiency is something that we have thought about a great deal. If necessary, we believe that we can achieve substantial storage savings with a couple of simple methods like those mentioned above.

As noted, file management tasks become much more complex as the number of data-compression techniques increases. Hence, we are not inclined to employ such techniques for MATH/PC, especially given the results of our benchmarks.

E. MAINTAINABILITY/OPERABILITY

One of the main reasons for moving the MATH model to the PC platform is to make the model more maintainable and easier to modify by programmers. The PC environment will make it much easier for programmers to edit the programs using a full-screen editor, to check for inefficient code, and to monitor the progress of the model. Since porting the model to the PC will require us to rewrite the code, we will be able to make use of the advances made possible by structured design and FORTRAN's new capabilities.

There are two tools currently touted by the PC industry as being breakthroughs in the area of maintainability and operability. The first one, also encouraged in the National Academy of Sciences report (Citro and Hanushek, 1991), is computer-aided software engineering (CASE). CASE technology is the automation of step-by-step methodologies for software and systems development from initial planning to maintenance. CASE tools help to relieve the drudgery of the development process and to standardize the design process. CASE tools are still relatively rare and most often found in large corporations. CASE tools are very expensive and are generally oriented toward transaction processing such as order-taking and inventory-tracking, which is not at all like the MATH model processing. The other tool is object-oriented programming (OOP). OOP tends to be graphics oriented, that is, displaying buttons and boxes on a screen and handling the user input. While we might want to use a package that uses OOP technology to develop screens, we are not likely to use it directly. While we do not expect to make use of these tools in the near future, they may one day provide us with powerful modeling techniques. Appendix D contains a detailed explanation of these tools.

F. FUTURE GROWTH

We have seen that PC technology has expanded rapidly in recent years. This might lead to the concern about whether our decisions will become obsolete, but we do not feel that it should be a major concern. We are confident that the options we have chosen will allow future growth, and that new technology will not make the PC-based MATH model obsolete. If anything, new technology will enable the model to run faster, handle larger files, and expand the scope and behavioral content of the model. We expect that in about two years, the model will run on a laptop if PC technology continues at the same rate. Also, new software tools such as profilers, CASE tools, and others will make modeling easier in general.

We feel we can safely say that five years from now, a MATH/PC model will run in under 10 minutes and take a relatively small part of the total disk space on a typical PC. We are not as sure

about which operating systems will be dominant, but we expect DOS to be in wide use. However, Windows NT, OS/2, or UNIX are distinct possibilities, and moving MATH/PC to them in the future would not be difficult.

VI. RECOMMENDED APPROACH

We have concluded that it is feasible and efficient to develop a CPS-based simulation capability on the PC. This technology is sufficiently advanced today to support simulation of the distributional effects of changes to the cash assistance and food stamp programs. We envision the model development process as having three phases. The first phase is this report, the second phase is the implementation of the recommended approach, and the third phase would be the completion of the PC model development. Clearly, we cannot achieve all the goals stated in Chapter IV within the resources allotted for Task 5.0. However, we can produce a complete and working model on the PC under the task. To ensure that we meet this essential goal, we will conduct Phase II--implementing a prototype PC-based model--in an incremental fashion so that the decision to develop a PC-based model can be reevaluated at key points in time. Below, we discuss the design of the PC-based model and how it interacts with the mainframe model. We next discuss the implementation plan and the planned incremental staging of that work.

A. MODEL DESIGN

We do not feel it is necessary or important to immediately implement the full capabilities of the

1. Scope

We have considerable latitude on the point where we split the system between the mainframe and the PC, but we recommend (as stated in Chapter IV) the following split to optimize the operation of two models in tandem:

- MATH/PC: Cash assistance eligibility and participation and food stamp eligibility and participation--both base-law and reform simulations
- MATH/Mainframe: Data conversion and basic recoding, aging, taxes, annual to monthly income allocation, and asset and expense imputation

The design and configuration of the MATH model remain unchanged in the plan. MATH/PC will consist of the following components:

- A module to coordinate the components of the model and the parameters
- A module to read the data
- A module to write the data
- Master routines for each of the following functions: cash assistance program unit formation, cash assistance eligibility and participation, and food stamp eligibility and participation
- A general-purpose master routine for ad hoc analysis

The read, write, and supervisory routines will be developed first for the PC, taking advantage of the capabilities of that platform, while the substantive models will be ported from the mainframe. Thus, from a substantive point of view, we will preserve the current capacity for reform simulation through changes in program parameters. (By current capacity, we refer to the model as it is now being implemented in building the new MATH database. In particular, we include the new public assistance module, which is more flexible and efficient than the old public assistance module.)

In designing this two-part system, we need to ensure that the two components of the model function as one. We will ensure this cohesion through the functional design itself and through the

design of the input/output routines. By including both cash assistance and food stamps on the same platform, we will not need to transfer between platforms in the middle of a quick response request since all components required to simulate the need-tested programs will be in one place. By ensuring that the mainframe model can write data in a universal format and that the PC model can convert that data to the MATH/PC format, we will have the required physical interface between the two models.

2. Operation

The MATH/PC model will be used to simulate the base-law program as well as reform-law programs. It will thus operate independently of the mainframe model once we have aged the data and simulated taxes. Specifically, we will use the MATH model as we currently do to convert a raw CPS to MATH format, to create basic definer variables, to age the data, to simulate taxes, and to impute assets and expenses. (This process will yield a base-law program file lacking only cash assistance and food stamp benefits.) We will then use the mainframe model to write the aged MATH file in a format suitable for transfer to the PC.

The MATH/PC model will read the file transferred from the mainframe and convert it to MATH/PC format. Next, we will simulate base-law cash assistance and food stamps, producing a full current services file. This file will be archived according to the procedures discussed in Section VI.B and used as a basis to generate a quick-response file for reform-law simulations. The quick-response file will be a MATH/PC file, and it may contain a subset of the observations (all households below 250 percent of poverty on an annual basis or who report cash assistance or food stamps on the raw file).[1]

---

[1]This step of reducing the file to a subset of households is not necessary either from a substantive point of view or from a processing or storage point of view, but it is preferable. We do achieve a reduction in processing time of a few minutes, which is desirable. More importantly, however, it gives us a smaller file to transport between PC computers, an important feature in the decentralized PC system.

Reform-law cash assistance and food stamp simulations will be carried out with MATH/PC in the following manner: the MATH/PC model will read the quick-response file created by the MATH/PC model; simulate the reform plan by using the base-law program model and parameters changed by the user where feasible, and modified code where not; produce summary tables displaying net effects and gainer/loser tables illustrating gross impacts of the reform; and produce an auxiliary output file consisting of record identifiers and the reform-law results for each household on the database.

B. IMPLEMENTATION

The tests and other investigations conducted thus far in Phase I have given us a good sense of how to implement the CPS-based modeling capabilities on the PC. In this section, we discuss implementation in terms of the recommended environment (hardware and software), the data management functions, the supervisory functions, the user interface, the documentation, and the archival and maintenance procedures.

1. Environment

Given the successful application of both the QC and SIPP-based simulation models on the PC, the speed of the FORTRAN applications tested, and the use of DOS by MPR and FNS, we will implement MATH/PC in a DOS environment and develop the modeling functions in FORTRAN.

Aside from its speed, FORTRAN has some additional advantages for our application. First, the existing MATH model is programmed in FORTRAN so that portions of the code can be transferred directly to the new platform and will not have to be rewritten. Second, FORTRAN compilers support structured programming techniques, have a scientific/engineering focus, and incorporate good debugging facilities. Third, FORTRAN compilers are fairly standardized, and thus with careful attention to coding in a "standard" way, we expect to be able to port the model to a different operating system with minimal effort. For several years, MPR has written and compiled applications

on the PC and then uploaded to the mainframe, with only minimal code changes required to compile and execute the application on the mainframe. We also had to change FORTRAN code in our DOS benchmark programs to get them to run on a UNIX machine.

At this time, we are also planning to implement the data management functions in FORTRAN. The tests conducted using this language indicate we can achieve highly acceptable speeds with certain database designs. However, we want to experiment further before making our final decision. Our alternate choice of computer language for implementing the data management functions is C, which we will use if the more complete test application of FORTRAN proves disappointing. As discussed subsequently, we will also use one or more existing database management systems (to be chosen later) to develop a user interface and to manage the documentation.

We will design the model such that it can execute on a 386 33MHz computer with a 387 MATH coprocessor, 4MB of RAM, and a 200MB hard disk drive. However, we expect to operate the model on a much faster machine for quick response activities to achieve good turnaround times. The preferred machine will be a 486 EISA computer with a built-in MATH coprocessor, 8MB of RAM, a 500MB hard disk drive, and a disk-caching controller. With this faster machine and a FORTRAN-based model, we expect to execute a job that includes five food stamp reform simulations in under one hour in the worst-case scenario.

We emphasize that it is extremely unlikely that FORTRAN will become obsolete as a language. While DOS may become obsolete, it will not happen soon, and we will still ensure that an easy migration path will be available. There are several companies--including Microsoft, Lahey, SVC, and Rylan McFarland--that sell FORTRAN compilers. Many of these companies are developing compilers that will meet the latest standard (FORTRAN 90), which will offer greater flexibility and many C-like extensions. As long as FORTRAN is supported by the PC community, there is no chance that MATH/PC will become obsolete. Indeed, we believe that as PC systems become more

powerful and more corporations downsize mainframe applications, FORTRAN as a PC language will become even more important.

2. Data Management

The Phase II model will incorporate some but not all of our desired data management features. We will design an internal format for the model according to the standards discussed earlier in the report. Furthermore, we will incorporate the facility to read and write any data file in that format. In other words, we will have the flexibility to process any number of records and any number of variables as long as they are created in the image of a MATH/PC file.

We are not yet ready to specify the precise format of a MATH/PC file. However, at a minimum we will design the system to read and write a household's worth of data at one time, retaining the full amount of household, family, and person information. This information will be loaded into arrays with linkage variables to allow easy movement among record types. The data will be stored in variable-length records, with information compressed in a manner that balances the need to minimize storage against the need for clear, efficient software. We will determine the final MATH/PC format after experimentation with the alternative approaches discussed in Chapter V. The ideal file format is one that can be read directly by almost any package such as SAS, TPL, or by a language such as C or COBOL. We will balance our desire for I/O efficiency with our desire for a direct interface with other applications.

In addition to the capability to read and write any file in a MATH/PC format, we will incorporate the ability to write small auxiliary files in any given MATH/PC run. The auxiliary file will typically contain the reform-law output variables from a quick-response run and variables needed to identify households, families, and persons.

We will not incorporate the ability to read two files in tandem or to read and write a character file directly by the model in this initial implementation. However, we will retain the ability to accomplish these goals using special-purpose software. In other words, we will be able to read a file

produced by the mainframe MATH model and convert it to a MATH/PC format, and we will be able to link an auxiliary file to a MATH/PC file. We just will not be able to implement these options in the most elegant, efficient, or preferred way in Phase II of this project.

Even though we do not plan to proceed directly to full implementation of the data management functions in Phase II, we will incorporate more flexibility in the initial MATH/PC model than we currently have with the mainframe MATH model. We will effectively eliminate restrictions on record sizes and on household sizes, we will minimize requirements for variables to be in fixed locations or perhaps drop them entirely, we will increase the clarity and maintainability of the data management software, and we will explore an automated link between the MATH/PC database and statistical packages like TPL or SAS.

3. Supervisor

We will reprogram the MATH SUPERVISOR and optimize it for the PC. In doing so, we have three goals. The first goal is to make this part of the model more modular and self-contained so that it will be easier to add new master routines as they become available on the PC and to add or delete parameters as the programs or the policy interests change. The second goal is to increase the parameterization of this routine so that as fundamental aspects of the data or model change (such as the individual's age, which distinguishes persons with a full complement of variables from persons with only an abbreviated set of demographic characteristics), the model can easily adapt. The third goal is to facilitate the new approach to the user interface (discussed in the next section). To be consistent with the new user interface design, we need to move the default for a module's parameters out of the SUPERVISOR and into the individual master routines.

4. User Interface

In Phase II of this project, we will establish the foundation for the new user interface and develop a prototype. For each module, we will develop a file that reflects all parameters of the

simulation, including those with substantive implications as well as those with processing implications. All parameters in the current model (that is, the mainframe model we are now updating) will be incorporated into the parameter files. Initially, these parameter files will be manually edited to set up MATH/PC jobs for execution on the PC.

We will develop a prototype of the system to provide a user interface for analysts and other nonprogrammers. The software will be intended to (1) facilitate access to the parameter files just described and their documentation; (2) provide a search facility to locate specific parameters for a specified master routine; and (3) create the ability to store and retrieve parameters established for previous runs, to program temporary variables through the use the of the general-purpose routine, to execute a MATH/PC job, and to link the MATH/PC results back to the run specification for archiving.

5. Documentation

In Phase II we will prepare comprehensive documentation for the PC-based model in machine-readable form, although we will not fully achieve all of our eventual goals outlined in Chapter III. First, the full design of the MATH codebook system will include not only the data dictionary and variable indices in machine-readable form, but also will include on-line access to the data dictionary and search facilities. The MATH and MATH/PC codebook will be one reference because all data items created by the MATH model are potentially part of the MATH/PC model. In other words, as in the current codebook, we will document every variable in the joint system, including those produced by both models and those originally extracted from the CPS. The document will describe the origin of the data element, its name, MATH code number, format, value ranges, and record type.

Second, we will expand the technical description to include a cross-reference and indices. We will also reorganize the document to clarify the scope and sequencing of each master routine. Finally, we will separate the current technical description into two parts, with one part devoted to the

MATH/PC model. We believe it will facilitate understanding of the content of the MATH/PC model if the user can go directly to the specifications of the modules in that system.

Third, we will develop a user's guide to the MATH/PC model that describes in a nontechnical way the scope of the model, its sequencing, the user interface, and the appropriate uses of the system. Given the developmental staging of the user interface discussed above, we do not expect to completely finish the guide in Phase II.

Fourth, we will devote considerable effort to documenting programming conventions in the user's guide. Much of the success of the MATH/PC model will hinge upon good archival and maintenance techniques, which involve a number of conventions for writing software; naming variables and routines; naming and referencing subroutines and master routines; and storing and accessing the model, the model output, and the run setups. With good coding techniques imposed on software developed for the MATH/PC, the code becomes as close to self-documenting as possible, an extremely desirable feature in this type of application.

Fifth, we should in the long term develop tools to create machine-readable codebooks for other packages, such as SAS and TPL, automatically from the internal codebook for both the entire file and extracts of the full file.

6. Archival and Maintenance

In deciding that now is the right time to implement CPS-based simulation capability on the PC, we recognize that there are still some weaknesses to operating on this platform--especially in the areas of archival and maintenance. While network backup is feasible, it is difficult to implement if the data being backed up is not on the file server. Since the MATH/PC file sizes and data-crunching requirements would overwhelm a typical file server, we do not expect to implement the full model on a network. The lack of centralized backup facilities on the PC presents a special challenge to model developers in achieving the requirements noted earlier.

There are some special features of the PC environment that will be particularly valuable to this aspect of the modeling effort, such as WORM disks, where we can essentially write the official model once and it cannot be overwritten. Other removable storage options abound: the Bernoulli disk with capacities up to 90MB (180MB with compression) and the erasable optical disk with capacities up to 1GB of data on a single cartridge. Optical disk drives alleviate storage concerns, although the drives are relatively expensive, and the cost per megabyte of storage is higher than that of tape backup. However, both optical disks and Bernoulli disks work like regular hard disk drives and are therefore much easier to use than tapes. We will hold our options open until we are able to test the hardware in Phase II.

In Phase II, we will develop a system of archival and maintenance that will consist of a series of procedures governing model and data access, modification, and storage. The procedures must be in place before we start using the PC-based model to simulate reforms to ensure that we do not accidently destroy the base-law program model or database, or misplace important reform-law simulation specification and output.

C. INCREMENTAL STAGING OF PHASE II

We are planning an incremental approach to the development of the MATH/PC model. Although we are confident that the PC platform is ready for a model of this type, we do not want to proceed too far without assessing the progress and likely outcomes. We discuss below the steps and present a schedule for the task.

1. Development Steps

The first step in Phase II will be to develop the programming conventions to be imposed on the MATH/PC software development effort. We estimate these will be completed within one month.

Second, we will adapt the food stamp module from the April 1994 model, the April 1994 database, and all of the supervisory and I/O routines needed to run the food stamp module. This

effort will involve adaptation of the I/O routines to process a PC file but will not involve a full-scale revamping of the SUPERVISOR. The purpose of this exercise will be to certify that our estimates of processing speed and our assessment of the appropriateness of the PC FORTRAN compiler are correct. We estimate this step will take three months to complete. At the completion of the conversion, we will re-execute a base-law program for food stamps, comparing the results of the MATH/PC model to the results of the mainframe model for every household on the database.

Third, assuming we are successful in completing steps 1 and 2, we will optimize the SUPERVISOR and the I/O routines for the PC. We estimate this step will take three months as well. At the completion of the optimization of the SUPERVISOR, we will repeat the food stamp base-law program simulation. We will also test the operation of the food stamp parameters by resetting each one sequentially and running reforms on a subset of households. We will repeat this on the same households on the mainframe.

The fourth step will be to transport the public assistance modules to the PC and adapt them to operate within the new MATH/PC SUPERVISOR. Given that we have just recently reprogrammed two of the three modules (UNIT 7 and PBLAST) on the PC, we anticipate that this step will require only one month to complete.[2] We will test the conversion of these modules in the same manner that we propose to test the conversion of the food stamp module. First, we will resimulate the base-law program comparing the microlevel output to the mainframe simulation. Second, we will execute the test of the operation of the parameters.

The fifth step in Phase II will be the adaptation of the MATH documentation to the MATH/PC model. As noted, we will make improvements in the accessibility and usability of the codebook, enhance the technical description of the model algorithms, develop a user's guide, and document the coding conventions developed in step 1. We estimate this will require about three months to complete.

---

[2]We developed the software, compiled it, and checked the syntax on the PC. However, the final phase of debugging was carried out on the mainframe computer.

The final step will be to develop the prototype user interface, which will also require about one month to complete.

2. Schedule

In September 1992, we presented the plan described in this report to FNS and to FNS-designated outside reviewers for comments. From October through December 1992, we will develop the coding conventions that will govern the remaining activities under this project and we will perform step 2 (including testing): the initial transfer of the food stamp model and the supervisory and I/O routines needed to execute the food stamp model. From December 1992 through February 1993, we will optimize the supervisory routines with testing to take place in March. In February 1993, we will convert the cash assistance modules and begin the documentation step to be completed in April 1993, when we will also develop the prototype user interface. Finally, in May 1993, we will present a demonstration of the MATH/PC model to FNS. It will include the results of system tests demonstrating the comparability of the mainframe and PC model estimates of cash assistance and food stamp base-law and reform programs. We will also illustrate the speed of operation of the new model. Finally, we will demonstrate the prototype user interface.

REFERENCES

Citro, Constance F., and Eric A. Hanushek. *Improving Information for Social Policy Decisions. The Uses of Microsimulation Modeling*, vol. 1: *Review and Recommendations*. Washington, DC: National Academy Press, 1991.

Chan, Stephan. A presentation by Stephan Chan of Uniprime Systems, Inc. concerning UNIX versus OS/2 versus DOS. July 1, 1992.

Doyle, Pat. "Food Stamp Program Participation Rates: August 1985." In *Current Perspectives on Food Stamp Program Participation*. Alexandria, VA: U.S. Department of Agriculture, Food and Nutrition Service, 1990.

Doyle, Pat, and Harold Beebout. "Food Stamp Program Participation Rates." In *Current Perspectives on Food Stamp Program Participation*. Alexandria, VA: U.S. Department of Agriculture, Food and Nutrition Service, 1988.

Doyle, Pat, Judy Richter, Ann Huff, and Carole Trippe. "The MATH Technical Description." Washington, DC: Mathematica Policy Research, Inc., 1990.

Doyle, Pat, Judy Richter, Richard Shin, and Carole Trippe. "Creation of the 1991 MATH Database for the March 1988 Current Population Survey." Washington, DC: Mathematica Policy Research, Inc., 1990.

Doyle, Pat, and Carol Trippe. "Food Stamp Program Participation Rates: January 1988." In *Current Perspectives on Food Stamp Program Participation*. Alexandria, VA: U.S. Department of Agriculture, Food and Nutrition Service, 1992.

Doyle, Pat, and Carole Trippe. "Food Stamp Program Participation Rates: January 1989." In *Current Perspectives on Food Stamp Program Participation*. Alexandria, VA: U.S. Department of Agriculture, Food and Nutrition Service, 1992.

Doyle, Pat, and Carole Trippe. "Improving the Income Allocation Procedures in MATH." Washington, DC: Mathematica Policy Research, Inc., 1991.

Faison, Ted. *Borland C++ 3 Object-Oriented Programming*. Carmel, IN: SAMS, 1992.

Fiedler, David. "The UNIX Operating System: Many Flavors, Many Forms." *Network Computing*, vol. 3, no. 5, May 1992, pp. 114-120.

Henle, Richard A., and Boris W. Kuvshinoff. *Desktop Computers*. Oxford University Press, 1992.

Lewis, Gordon H., and Richard C. Michel. *Microsimulation Techniques for Tax and Transfer Analysis*. Washington, DC: The Urban Institute Press, 1990.

Machrone, William, and Michael J. Miller. "Eighth Annual Awards for Technical Excellence." *PC Magazine*, vol. 10, no. 22, December 1991, p. 122.

Mendelson, Edward. "Premium Insurance Backup Software Gets Better." *PC Magazine*, vol. 10, no. 11, June 1991, p. 103.

Rinaldi, Damian, and William Gannon Jr. "Characteristics for CASE Success." *CASE Products Guide/1992.* Westborough, MA: Sentry Publishing Company, 1991.

Seymour, Jim. "Platforms: How the PC Stacks Up." *PC Magazine*, vol. 10, no. 10, May 1992, p. 113.

Sprague, Christopher. "Managing CASE Tools: It's More Than Just Using Them." *CASE Product Guide/1992.* Westborough, MA: Sentry Publishing Company, 1991.

Statistics Canada. "SPSD/M Introductory Manual." December 20, 1990.

Webb, Randall. Personal communication concerning TRIM2 design. April 12, 1992.

Williamson, Mickey. "GUI: The Next Challenge." *CASE Strategies*, vol. 4, no. 5, May 1992, p. 2.

Williamson, Mickey. "Process Metrics: More Talk than Action, Expert Says." *CASE Strategies*, vol. 4, no. 6, June 1992, p. 15.

Williamson, Mickey. "U.S. Government Defines Software Engineering Environment." *CASE Strategies*, vol. 4, no. 6, June 1992, p. 5.

Table of Contents

APPENDIX A

GLOSSARY

GLOSSARY

Array: A collection of similar data elements, all sharing the same variable name; individual elements are referenced by subscripts.

Artificial Intelligence Programming: The use of methods to develop a machine that can improve its own operations or can perform functions such as reasoning or learning.

ASCII: Abbreviation for American Standard Code for Information Interchange, an eight-level code(seven bits plus parity check) widely used for data representation in data processing systems.

Average access time: The average amount of time that it takes the hard disk to position the drive heads prior to reading or writing data.

BASIC: Acronym for beginners' all-purpose symbolic instruction code. BASIC is a conversational type of programming language using English-like statements and mathematical notation.

Batch mode: A computer mode in which programs are submitted for execution and the results returned at a later time.

Baud: Unit of modulation. Rate at which carrier containing transmitting signal changes.

Benchmark: An established point of reference against which computers or programs can be measured in tests comparing their performance, reliability, etc.

Bernoulli Cartridge: A removable cartridge that holds a special 5 1/4-inch flexible disk inside a rigid case and provides high-speed, high-capacity data storage. Bernoulli cartridges come in 20, 44, and 90MB capacities.

Bit: The smallest element of binary data. It can only be 0 or 1.

Boolean variables: Variables that can only have one of two values - true or false, on or off, 0 or 1.

BPI: Bits per inch. 6250 BPI is the standard recording density for magnetic tape on 10-inch reels. It is the number of bits which can be written to a single track in an inch of tape. On a 9-track tape it is equivalent to 6250 bytes per inch (8 bits for data and 1 bit for error checking).

Byte: 8 bits. Byte indicates the numer of consecutive binary digits that are usually operated on as a unit. The value of one byte can range from 0 to 255.

C: A high-level language developed in the early 1970s. It is a general purpose language that is modular oriented and allows the programmer to create additional commands called functions.

C++: Object-oriented version of C language.

Cache: Area of fast, static RAM that keeps data and instructions that are most used by the CPU. If data is in cache memory, it can be delivered to the CPU with no delay.

CASE: Computer-aided systems engineering tools. CASE technology is the automation of step-by-step methodologies for software and systems development from intial planning to ongoing maintenance.

CD-ROM: Acronym for compact disk read-only memory. Disk storage that is read-only and is used to store general-purpose digital data.

Character user interface (CUI): A character prompt appearing on the screen, asking for information from the user.

Chip: A silicon slab with added impurities so that circuit path and device interconnections are formed within the structure. The CPU resides on a very large-scale integrated chip which also holds other electronics including a control unit and an arithmetic logic unit.

Clipper: A programming language with a powerful user interface and database processing commands.

COBOL: Acronym for common business oriented language. COBOL makes use of English-like

EISA: Extended industry standard architecture. A relatively new data bus design that is 32 bits wide and can operate much faster that the older ISA bus.

Extended memory: Directly addressable, executable memory beyond one megabyte.

Floppy disks: A magnetic storage medium that uses flexible disks that provide random access storage for 300,000 or more bytes.

FORTRAN: Acronym for formula translator, one of the most widely used languages for scientific and business problems.

Full-screen editor: Text editor that allows full-screen display and cursor movement for editing.

Gigabyte (GB): One billion bytes. This is equivalent to the data that can be contained on six reels of 6250 BPI tape.

Graphical user interface (GUI): Picture-oriented shell that features icons and pull-down menus providing the user simple and obvious access to computer files, programs, and functions.

IBM 370: Large IBM mainframe computer. Designed as a batch system that can run on-line jobs.

IBM MVS: Multiple virtual system. A multiprogramming system for larger IBM computers.

IBM PC/XT/AT: The original family of IBM personal computers. The original PC series was a floppy disk machine with the 8088 microprocessor. The XT was the hard disk version, and the AT (advanced technology) series is based on the 80286 microprocessor.

IDE: The integrated drive electronics interface integrates both a hard disk and its controller in a single unit.

Intel 8088: Early microprocessor by Intel with 16-bit data paths internally, but brought out only eight data lines. Original IBM PC and XT were based on the 8088 microprocessor.

Intel 80286: Microprocessor by Intel with 16-bit data path and 16 data lines.

Intel 80386: Microprocessor by Intel with full 32-bit data path and 32-bit direct addressability.

Intel 80486: Microprocessor by Intel that is similar to the 80386 but includes a built-in math coprocessor and a built-in 8K data cache that doubles throughput at the same clock speed.

ISA: Industry standard architecture. The 16-bit wide data bus design originally introduced with the IBM-AT.

I/O: Input/output. Anything to do with the reading and writing of data.

JCL: Job control language. A language specifically used to code job control statements.

Kilobyte (K or KB): One thousand bytes. DOS memory is often described in kilobytes as are older floppy sizes.

LAHEY: A FORTRAN compiler.

LISP: Acronym for list processing, a language designed for the handling of lists and recursive data.

Local Area Network (LAN): An in-house data communications system, usually within a single building, that connects a number of microcomputers together. This permits sharing of devices and files.

Lotus 1-2-3: A popular spreadsheet program. 1-2-3 combines a spreadsheet with simple information management and analytical graphics.

Magnetic optical (MO) storage: Erasable optical disk that can be rewritten many times. The magneto-optic method writes using a strong magnetic field to modify the crystal pattern on the optical disk.

Math coprocessor: Special-purpose chip that supplements the CPU by taking over math functions.

Megabyte (MB): One million bytes. Hard disk capacity is usually measured in megabytes.

Megahertz (MHz): One million cycles per second. Computers rely on the frequency of the clock. This is related to the measure of how many instructions a computer may process, but it is not directly translatable. For example, older PCs would take several cycles to execute an instruction, whereas newer CPUs can execute one or more instructions per cycle.

Microprocessor: The computer's central processing unit on a semiconductor chip. Through large-scale integration technology, the microprocessor is able to hold huge numbers of tiny electronic circuits that provide the means for executing instructions that control the operation of the computer.

Milliseconds (ms): Thousandths of a second.

MIPS: Acronym for millions of instructions per second, a measure of CPU speed.

Modem: An electronic device used to connect computers and terminals over telephone circuits. A modem converts the computer's digital signals to analog signals that can be transmitted on phone lines and then reconverts back to digital any signals that have been received.

MS-DOS: A general purpose disk operating system that is used with IBM PC and compatible computers.

MS-WINDOWS: A graphical environment intended to enhance the existing MS-DOS operating system. Windows allows multitasking and has advanced memory management features.

Multiprogramming: The concurrent execution of two or more programs.

Multitasking: The procedures in which several separate but interrelated tasks operate within a single program identity. Multitasking differs from multiprogrammming in that common routines and disk files may be used.

Network: Any system of multiple interrelated computers.

The NORTON Utilities: A package of utility programs for the PC. These utilities allow the user to protect data by recovering files, to browse files, to label disks, to print files, etc.

Object-oriented programming (OOP): Programming style in which the top level is the collection of objects (or data elements) and operation is secondary.

Operating system (OS): A basic group of programs for supervising processing operations.

Optical disk: Disk storage whose main recording method consists of writing with a laser and then reading the laser light reflections.

OS/2: An operating system jointly developed by IBM and Microsoft.

Pascal: A structured, high-level language used in a wide variety of applications.

Porting: Moving a program to a different computer environment.

POSIX: Portable operating system interface for UNIX.

P5 (or 80586): Unofficial name for the next-generation computer chip being developed by Intel. It is expected to be four or five times faster than the 80486 chip.

QIC: A tape specification for 1/4" tapes about the size of an audio cassette.

RAM: Acronym for random-access memory. RAM is read/write memory, where programs and data are executed.

ROM: Acronym for read-only memory, a memory that can not be altered by computer instructions.

SAS: Acronym for statistical analysis system, a software package designed for data analysis, programming, and statistical analysis.

SCSI: Small computer system interface, a general-purpose interface to handle details of data transfer.

Software Development Kit: Kit for developing software applications for Windows.

Stack: A portion of memory in which values are stored in much the same way as trays are stacked in a cafeteria. The last tray, or value, on the stack is the first to come off.

Text format: The mode in which letters and/or numbers are displayed in the form of conventional text for reading.

TPL: Acronym for a table-producing language that makes it easy to produce cross-tabulations.

UNIX: An operating system developed by Bell Laboratories for minicomputers and recently adapted for mainframe computers. UNIX permits several programs to run concurrently.

Visual BASIC: Version of BASIC for Windows. Allows easy development of menus and interactive programs.

WATCOM: A FORTRAN compiler.

Word: The size of the data that occupy one storage location and that are treated as a unit. This size varies from machine to machine: a word is 2 bytes on older PCs and 4 bytes (32 bits) on the 80386 and 80486 chips.

Workstation: A complete computing environment that is the combination of high-performance hardware, a mature operating system, well-connected applications software, and connectivity with other workstations.

WORM disks: Acronym for write-once-read-many, refers to a type of optical disk that can be written only once and cannot be erased or formatted.

8088, 80286, 80386, 80486, 80586: Microprocessors designed by Intel around which IBM has based its line of PCs. (See Intel 8088, etc.) (80586: See P5)

Table of Contents

APPENDIX B

SAMPLE TABLES FROM THE MATH MODEL

AVERAGE VALUES AND % WITH DEDUCTIONS AMONG ELIGIBLE FSTAMP HOUSEHOLDS.
AT EACH LEVEL OF AGREGATION THE DEDUCTIONS TABBED WERE CONSTRAINED TO BE LESS THAN GROSS MONTHLY INCOME.

	MONTH/NTH	CHLDCD	SHLTRD	MEDD	NOT USED	NOT USED	IMPUTED	COMPUTED	TOTAL
AVERAGE AMOUNTS									
	1	151.93	133.82	77.52	0.00	0.00	142.98	163.54	253.60
	2	155.53	133.98	77.27	0.00	0.00	144.18	185.33	276.55
	3	159.26	137.97	76.78	0.00	0.00	151.58	407.56	506.57
% WITH DEDUCTION									
	1	4.11	59.43	9.05	0.00	0.00	64.53	92.86	92.86
	2	4.52	59.85	8.81	0.00	0.00	64.86	93.08	93.08
	3	6.75	62.69	7.44	0.00	0.00	67.66	94.47	94.47

AVERAGE VALUES AND % WITH DEDUCTIONS AMONG PARTICIPATING FSTAMP HOUSEHOLDS.
AT EACH LEVEL OF AGREGATION THE DEDUCTIONS TABBED WERE CONSTRAINED TO BE LESS THAN GROSS MONTHLY INCOME.

	MONTH/NTH	CHLDCD	SHLTRD	MEDD	NOT USED	NOT USED	IMPUTED	COMPUTED	TOTAL
AVERAGE AMOUNTS									
	1	125.14	130.61	69.74	0.00	0.00	135.16	132.74	219.56
	2	134.29	130.49	69.74	0.00	0.00	135.82	142.04	229.53
	3	138.35	134.51	70.34	0.00	0.00	140.71	208.74	297.52
% WITH DEDUCTION									
	1	2.76	65.81	2.81	0.00	0.00	67.37	90.70	90.70
	2	2.97	66.19	2.78	0.00	0.00	67.73	90.81	90.81
	3	3.33	67.37	2.71	0.00	0.00	68.83	91.31	91.31

* NOTE THE AVERAGE VALUES ARE FOR HOUSEHOLDS WITH DEDUCTIONS

TABLE 2: SUMMARY STATISICS FOR PARTICIPATING HOUSEHOLDS
(COUNTS ARE IN THOUSANDS)

MTH/ MONTH	HOUSEHOLDS WITH ZERO NET INCOME	HOUSEHOLDS RECEIVING MINIMUM BONUS	HOUSEHOLDS W/ CHILDREN < AGE 18	HOUSEHOLDS W/CHILDREN AGE 5 TO 17	HOUSEHOLDS RECEIVING SSI	AVERAGE GROSS MONTHLY INCOME	AVERAGE NET MONTHLY INCOME	TOTAL HOUSEHOLDS
1	2089229.	201685.	4020543.	3242007.	1809642.	440.	240.	7479119.
2	2122722.	200033.	4100030.	3305513.	1817905.	451.	242.	7573107.
3	2416883.	197988.	4456341.	3584511.	1876334.	509.	236.	8002248.

PLAN26: ELIMINATE GROSS INCOME TEST

FOOD STAMPS
COUNT OF FILING UNITS RECEIVING FOOD STAMPS FOR: MONTH = APRIL NTH = 1
***** ELIGIBLES *****

UNIT'S INCOME (GMINC)	UNIT SIZE 1	2	3	4	5	6	7	8+	TOTAL UNITS	TOTAL BONUS VALUE
< OR = 0	436276.	196593.	122079.	112757.	46299.	24446.	7281.	4140.	949871.	192667607.
1- 99	271160.	119948.	60933.	20125.	21473.	3387.	0.	0.	497026.	85080073.
100-199	327091.	156862.	76348.	35481.	17029.	2313.	0.	0.	615125.	104064249.
200-299	398446.	252733.	144413.	83335.	42447.	19907.	6517.	4794.	952593.	180414135.
300-399	824198.	244075.	186360.	80518.	27114.	8813.	10816.	0.	1381893.	179116224.
400-499	1666367.	406130.	196511.	165216.	65797.	27118.	10134.	7812.	2545086.	240647198.
500-599	1174732.	404197.	241000.	90797.	47923.	35152.	0.	5494.	1999296.	170206814.
600-799	686682.	822815.	330657.	235453.	144720.	45641.	18473.	29357.	2313798.	232383400.
800-999	166483.	522554.	238361.	224262.	139627.	71746.	26522.	16898.	1406452.	147526585.
1000 UP	13633.	136935.	371056.	553859.	326177.	215276.	98126.	125781.	1840842.	211259651.
TOTAL	5965067.	3262843.	1967718.	1601802.	878605.	453800.	177869.	194275.	14501981.	1743365936.
PEOPLE	5965067.	6525686.	5903152.	6407209.	4393025.	2722799.	1245085.	1735669.	34897692.	0.

NUMBER OF HHLDS WITH AFDC OR GA: 3704548.
NUMBER OF HHLDS WITHOUT AFDC OR GA : 10797433.

***** PARTICIPANTS *****

UNIT'S INCOME (GMINC)	UNIT SIZE 1	2	3	4	5	6	7	8+	TOTAL UNITS	TOTAL BONUS VALUE
< OR = 0	263092.	72121.	122079.	112757.	46299.	24446.	7281.	4140.	652214.	151058860.
1- 99	174586.	46316.	60933.	20125.	21473.	3387.	0.	0.	326819.	61142778.
100-199	251733.	114533.	76348.	35481.	17029.	2313.	0.	0.	497437.	88413190.
200-299	269859.	247937.	144413.	83335.	42447.	19907.	6517.	4794.	819210.	168454043.
300-399	481037.	244075.	186360.	80518.	27114.	8813.	10816.	0.	1038733.	157074203.
400-499	873518.	385429.	196511.	165216.	65797.	27118.	10134.	7812.	1731535.	212504274.
500-599	67873.	264170.	241000.	90797.	47923.	35152.	0.	5494.	752410.	126792977.
600-799	5396.	208478.	290470.	211068.	136140.	35562.	14150.	22767.	924030.	169100051.
800-999	2292.	0.	93450.	130564.	77064.	42032.	8770.	9856.	364028.	61582495.
1000 UP	2518.	0.	42819.	106093.	74980.	48064.	23341.	74888.	372704.	58826826.
TOTAL	2391904.	1583058.	1454383.	1035953.	556265.	246794.	81010.	129752.	7479119.	1254949696.
PEOPLE	2391904.	3166116.	4363148.	4143811.	2781325.	1480767.	567067.	1158836.	20052974.	0.

NUMBER OF HHLDS WITH AFDC OR GA: 3534654.
NUMBER OF HHLDS WITHOUT AFDC OR GA : 3944466.

PLAN 27: ELIMINATE GROSS INCOME TEST & EARNED DED = 25%

FOOD STAMPS
COUNT OF FILING UNITS RECEIVING FOOD STAMPS FOR: MONTH = APRIL NTH = 2
***** ELIGIBLES *****

UNIT'S INCOME (GMINC)	UNIT SIZE 1	2	3	4	5	6	7	8+	TOTAL UNITS	TOTAL BONUS VALUE
< OR = 0	436276.	196593.	122079.	112757.	46299.	24446.	7281.	4140.	949871.	192667607.
1- 99	271160.	119948.	60933.	20125.	21473.	3387.	0.	0.	497026.	85080073.
100-199	327091.	156862.	76348.	35481.	17029.	2313.	0.	0.	615125.	104072795.
200-299	398446.	252733.	144413.	83335.	42447.	19907.	6517.	4794.	952593.	180509426.
300-399	824198.	244075.	186360.	80518.	27114.	8813.	10816.	0.	1381893.	179617780.
400-499	1666367.	406130.	196511.	165216.	65797.	27118.	10134.	7812.	2545086.	242100546.
500-599	1174732.	404197.	241000.	90797.	47923.	35152.	0.	5494.	1999296.	172837838.
600-799	687504.	822815.	330657.	235453.	144720.	45641.	18473.	29357.	2314619.	238960676.
800-999	253852.	529833.	238361.	224262.	139627.	71746.	26522.	16898.	1501101.	157283123.
1000 UP	24352.	246177.	460047.	641917.	370905.	234645.	101430.	132095.	2211567.	253274510.
TOTAL	6063978.	3379365.	2056709.	1689860.	923333.	473168.	181173.	200590.	14968176.	1806404373.
PEOPLE	6063978.	6758729.	6170125.	6759439.	4616664.	2839009.	1268213.	1800971.	36277128.	0.

NUMBER OF HHLDS WITH AFDC OR GA: 3724808.
NUMBER OF HHLDS WITHOUT AFDC OR GA : 11243368.

***** PARTICIPANTS *****

UNIT'S INCOME (GMINC)	UNIT SIZE 1	2	3	4	5	6	7	8+	TOTAL UNITS	TOTAL BONUS VALUE
< OR = 0	263092.	72121.	122079.	112757.	46299.	24446.	7281.	4140.	652214.	151058860.
1- 99	174586.	46316.	60933.	20125.	21473.	3387.	0.	0.	326819.	61142778.
100-199	251733.	114533.	76348.	35481.	17029.	2313.	0.	0.	497437.	88421737.
200-299	269859.	247937.	144413.	83335.	42447.	19907.	6517.	4794.	819210.	168549334.
300-399	481037.	244075.	186360.	80518.	27114.	8813.	10816.	0.	1038733.	157570570.
400-499	873518.	385429.	196511.	165216.	65797.	27118.	10134.	7812.	1731535.	213909944.
500-599	67873.	264170.	241000.	90797.	47923.	35152.	0.	5494.	752410.	127997542.
600-799	7058.	212392.	290470.	211068.	136140.	35562.	14150.	22767.	929605.	172669275.
800-999	2292.	831.	98545.	130564.	79834.	42032.	10974.	9856.	374928.	65493479.
1000 UP	2518.	0.	56426.	124345.	95660.	65565.	29463.	76239.	450216.	72523316.
TOTAL	2393566.	1587803.	1473084.	1054206.	579715.	264296.	89335.	131103.	7573107.	1279336835.
PEOPLE	2393566.	3175605.	4419250.	4216821.	2898574.	1585773.	625346.	1179108.	20494043.	0.

NUMBER OF HHLDS WITH AFDC OR GA: 3554592.
NUMBER OF HHLDS WITHOUT AFDC OR GA : 4018515.

Table of Contents

COMPARISON OF "CURRENT LAW" PLAN FSBON001 WITH "PROPOSED LAW" PLAN FSBON027.
THIS PAGE TABS HOUSEHOLDS WITH EARNINGS
ENTRIES ARE % OF CURRENT LAW AMOUNTS. EACH CELL SHOWS COUNTS OF: HOUSEHOLDS
PEOPLE WITH EARNINGS
$ REDUCTION UNDER PROPOSED LAW

FSTAMP PARTICIPATION STATUS			% OF POVERTY							
FSBON001	FSBON027		0 INCOME	1-49%	50-99%	100-149%	150-199%	200-249%	250%+	TOTAL
PARTIC.	INELIG.		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	ELG,NPT		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. $51+	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. 21-50	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. 10-20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. $2-9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	NOCHANGE	100.000	53.115	9.498	6.633	0.000	0.000	0.000	18.534
			100.000	52.957	9.075	7.220	0.000	0.000	0.000	18.095
			0.000	-0.018	-0.014	-0.011	0.000	0.000	0.000	-0.015
PARTIC.	PARTIC.	G. $2-20	0.000	46.885	88.706	76.492	54.717	0.000	0.000	77.175
			0.000	47.043	88.602	72.776	53.038	0.000	0.000	76.586
			0.000	-1.064	-4.891	-8.681	-5.660	0.000	0.000	-3.979
PARTIC.	PARTIC.	G. 21-50	0.000	0.000	1.797	16.875	45.283	0.000	0.000	4.291
			0.000	0.000	2.322	20.004	46.962	0.000	0.000	5.318
			0.000	0.000	-0.240	-4.246	-13.310	0.000	0.000	-0.587
PARTIC.	PARTIC.	G. $51+	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
NPARTIC.	PARTIC.		0.000	0.453	2.807	29.504	103.521	2933.459	0.000	7.854
			0.000	0.440	3.611	30.974	130.231	2933.459	0.000	9.097
			0.000	-1.435	-4.281	-23.214	-48.131	-61602.641	0.000	-5.286
AMOUNT UNDER CURRENT LAW			1354.	298142.	831774.	222956.	13531.	0.	0.	1367756.
			1354.	307188.	870462.	249705.	13959.	0.	0.	1442668.
			368292.	82001781.	146386461.	22273318.	1385364.	0.	0.	252415214.

COMPARISON OF "CURRENT LAW" PLAN FSBON001 WITH "PROPOSED LAW" PLAN FSBON027.
THIS PAGE TABS HOUSEHOLDS WITH AGED MEMBERS
ENTRIES ARE % OF CURRENT LAW AMOUNTS. EACH CELL SHOWS COUNTS OF: HOUSEHOLDS
AGED PEOPLE
$ REDUCTION UNDER PROPOSED LAW

FSTAMP PARTICIPATION STATUS			% OF POVERTY						
FSBON001	FSBON027		0 INCOME	1-49%	50-99%	100-149%	150-199%	200-249%	250%+	TOTAL
PARTIC.	INELIG.		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	ELG,NPT		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. $51+	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. 21-50	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
.			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. 10-20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. $2-9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	NOCHANGE	100.000	96.124	96.452	64.871	0.000	0.000	0.000	95.459
			100.000	96.582	96.144	65.670	0.000	0.000	0.000	94.939
			0.000	-0.006	-0.002	0.000	0.000	0.000	0.000	-0.004
PARTIC.	PARTIC.	G. $2-20	0.000	3.876	3.548	33.308	100.000	0.000	0.000	4.487
			0.000	3.418	3.856	33.245	100.000	0.000	0.000	5.016
			0.000	-0.126	-0.295	-8.803	-135.714	0.000	0.000	-0.338
PARTIC.	PARTIC.	G. 21-50	0.000	0.000	0.000	1.821	0.000	0.000	0.000	0.054
			0.000	0.000	0.000	1.086	0.000	0.000	0.000	0.045
			0.000	0.000	0.000	-1.415	0.000	0.000	0.000	-0.016
PARTIC.	PARTIC.	G. $51+	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
NPARTIC.	PARTIC.		0.000	0.000	0.000	12.168	0.000	0.000	0.000	0.364
			0.000	0.000	0.000	7.254	0.000	0.000	0.000	0.304
			0.000	0.000	0.000	-23.435	0.000	0.000	0.000	-0.262
AMOUNT UNDER CURRENT LAW			18691.	300319.	1590862.	58909.	664.	0.	0.	1969445.
			20806.	340601.	1899723.	98819.	664.	0.	0.	2360612.
			2628796.	59122705.	119510382.	2047247.	9301.	0.	0.	183318432.

COMPARISON OF "CURRENT LAW" PLAN FSBON001 WITH "PROPOSED LAW" PLAN FSBON027.
THIS PAGE TABS HOUSEHOLDS WITH FEMALE HEADS WITH KIDS
ENTRIES ARE % OF CURRENT LAW AMOUNTS. EACH CELL SHOWS COUNTS OF: HOUSEHOLDS
PEOPLE IN THE HOUSEHOLD
$ REDUCTION UNDER PROPOSED LAW

FSTAMP PARTICIPATION STATUS			% OF POVERTY						
FSBON001	FSBON027		0 INCOME	1-49%	50-99%	100-149%	150-199%	200-249%	250%+	TOTAL
PARTIC.	INELIG.		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	ELG,NPT		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. $51+	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. 21-50	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. 10-20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. $2-9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	NOCHANGE	100.000	96.141	73.282	17.757	0.000	0.000	0.000	82.891
			100.000	95.335	70.949	14.301	0.000	0.000	0.000	81.176
			0.000	-0.001	-0.002	-0.019	0.000	0.000	0.000	-0.002
PARTIC.	PARTIC.	G. $2-20	0.000	3.859	26.704	63.422	0.000	0.000	0.000	16.209
			0.000	4.665	29.031	61.486	0.000	0.000	0.000	17.609
			0.000	-0.078	-1.560	-7.971	0.000	0.000	0.000	-0.728
PARTIC.	PARTIC.	G. 21-50	0.000	0.000	0.014	18.821	100.000	0.000	0.000	0.900
			0.000	0.000	0.020	24.212	100.000	0.000	0.000	1.214
			0.000	0.000	-0.002	-4.767	-33.465	0.000	0.000	-0.102
PARTIC.	PARTIC.	G. $51+	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
NPARTIC.	PARTIC.		0.000	0.100	0.166	19.000	224.068	0.000	0.000	1.122
			0.000	0.410	0.275	20.305	203.231	0.000	0.000	1.432
			0.000	-0.311	-0.245	-15.432	-140.445	0.000	0.000	-0.597
AMOUNT UNDER CURRENT LAW			164420.	1347165.	1276438.	125898.	2379.	0.	0.	2916300.
			574702.	4946358.	4627777.	489234.	9945.	0.	0.	10648016.
			50546515.	378329700.	227469654.	12889273.	193084.	0.	0.	669428226.

COMPARISON OF "CURRENT LAW" PLAN FSBON001 WITH "PROPOSED LAW" PLAN FSBON027.
THIS PAGE TABS HOUSEHOLDS WITH ALL AGED OR DISABLED
ENTRIES ARE % OF CURRENT LAW AMOUNTS. EACH CELL SHOWS COUNTS OF: HOUSEHOLDS
PEOPLE IN THE HOUSEHOLD
$ REDUCTION UNDER PROPOSED LAW

FSTAMP PARTICIPATION STATUS			% OF POVERTY						
FSBON001	FSBON027		0 INCOME	1-49%	50-99%	100-149%	150-199%	200-249%	250%+	TOTAL
PARTIC.	INELIG.		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	ELG,NPT		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. $51+	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. 21-50	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. 10-20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. $2-9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	NOCHANGE	100.000	99.409	99.439	85.023	100.000	0.000	100.000	99.091
			100.000	99.487	99.168	77.556	100.000	0.000	100.000	98.354
			0.000	-0.009	-0.003	0.000	0.000	0.000	0.000	-0.004
PARTIC.	PARTIC.	G. $2-20	0.000	0.591	0.561	14.977	0.000	0.000	0.000	0.909
			0.000	0.513	0.832	22.444	0.000	0.000	0.000	1.646
			0.000	-0.016	-0.051	-3.717	0.000	0.000	0.000	-0.075
PARTIC.	PARTIC.	G. 21-50	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	G. $51+	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
NPARTIC.	PARTIC.		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
AMOUNT UNDER CURRENT LAW			47952.	268366.	1412136.	44395.	4485.	0.	325.	1777658.
			49809.	309417.	1727735.	88874.	4485.	0.	325.	2180645.
			5100581.	29168290.	74550219.	1073251.	44848.	0.	3251.	109940439.

COMPARISON OF "CURRENT LAW" PLAN FSBON001 WITH "PROPOSED LAW" PLAN FSBON027.
THIS PAGE TABS HOUSEHOLDS WITH EARNINGS, AGED, FEM. HEAD, OR ALL AGEDIS
ENTRIES ARE % OF CURRENT LAW AMOUNTS. EACH CELL SHOWS COUNTS OF: HOUSEHOLDS
PEOPLE IN THE HOSEHOLD
$ REDUCTION UNDER PROPOSED LAW

FSTAMP PARTICIPATION STATUS			% OF POVERTY							
FSBON001	FSBON027		0 INCOME	1-49%	50-99%	100-149%	150-199%	200-249%	250%+	TOTAL
PARTIC.	INELIG.		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	ELG,NPT		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. $51+	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. 21-50	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. 10-20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. $2-9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	NOCHANGE	100.000	92.726	77.853	27.884	24.894	0.000	100.000	80.943
			100.000	90.764	69.400	19.554	10.105	0.000	100.000	75.779
			0.000	-0.003	-0.005	-0.010	0.000	0.000	0.000	-0.004
PARTIC.	PARTIC.	G. $2-20	0.000	7.274	21.707	59.082	41.095	0.000	0.000	18.053
			0.000	9.236	29.382	60.396	37.109	0.000	0.000	22.282
			0.000	-0.179	-1.760	-7.943	-5.482	0.000	0.000	-1.029
PARTIC.	PARTIC.	G. 21-50	0.000	0.000	0.440	13.034	34.010	0.000	0.000	1.004
			0.000	0.000	1.219	20.050	52.785	0.000	0.000	1.939
			0.000	0.000	-0.086	-3.884	-12.892	0.000	0.000	-0.152
PARTIC.	PARTIC.	G. $51+	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
NPARTIC.	PARTIC.		0.000	0.070	0.687	22.789	77.750	2933.459	0.000	1.837
			0.000	0.324	1.517	28.621	138.708	11733.836	0.000	3.024
			0.000	-0.242	-1.540	-21.240	-46.622	-61602.641	0.000	-1.367
AMOUNT UNDER CURRENT LAW			219050.	1921804.	3399019.	288655.	18015.	0.	325.	5846868.
			640856.	6257914.	8342065.	945678.	44380.	0.	325.	16231218.
			57154541.	486137483.	406871169.	24344294.	1430212.	0.	3251.	975940949.

Table of Contents

COMPARISON OF "CURRENT LAW" PLAN FSBON001 WITH "PROPOSED LAW" PLAN FSBON027.
THIS PAGE TABS ALL HOUSEHOLDS
ENTRIES ARE % OF CURRENT LAW AMOUNTS. EACH CELL SHOWS COUNTS OF: HOUSEHOLDS
PEOPLE IN THE HOUSEHOLD
$ REDUCTION UNDER PROPOSED LAW

FSTAMP PARTICIPATION STATUS			% OF POVERTY							
FSBON001	FSBON027		0 INCOME	1-49%	50-99%	100-149%	150-199%	200-249%	250%+	TOTAL
PARTIC.	INELIG.		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	ELG,NPT		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. $51+	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. 21-50	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. 10-20	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	L. $2-9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
			0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC.	NOCHANGE	100.000	94.639	80.629	33.847	24.894	0.000	100.000	85.075
			100.000	92.633	73.058	24.522	10.105	0.000	100.000	80.350
			0.000	-0.002	-0.004	-0.010	0.000	0.000	0.000	-0.003

Table of Contents

COMPARISON OF "CURRENT LAW" PLAN FSBON001 WITH "PROPOSED LAW" PLAN FSBON027.
THIS PAGE TABS ALL HOUSEHOLDS BY REGION
ENTRIES ARE % OF CURRENT LAW AMOUNTS. EACH CELL SHOWS COUNTS OF: HOUSEHOLDS
PEOPLE IN THE HOUSEHOLD
$ REDUCTION UNDER PROPOSED LAW

FSTAMP PARTICIPATION STATUS		REGION				
FSBON001	FSBON027	NORTHEAST	NORTH CENTRAL	SOUTH	WEST	TOTAL
PARTIC.	INELIG.	0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000
PARTIC.	ELG,NPT	0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC. L. $51+	0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC. L. 21-50	0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC. L. 10-20	0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC. L. $2-9	0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000
PARTIC.	PARTIC. NOCHANGE	88.734	84.820	84.277	82.868	85.075
		86.343	81.348	77.650	78.964	80.350
		-0.004	-0.003	-0.001	-0.008	-0.003
PARTIC.	PARTIC. G. $2-20	10.802	14.484	14.865	15.984	14.139
		12.802	17.320	20.448	19.140	18.077
		-0.681	-0.751	-0.808	-1.001	-0.801
PARTIC.	PARTIC. G. 21-50	0.464	0.696	0.858	1.148	0.786
		0.855	1.332	1.902	1.896	1.573
		-0.078	-0.093	-0.128	-0.176	-0.118
PARTIC.	PARTIC. G. $51+	0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000
		0.000	0.000	0.000	0.000	0.000
NPARTIC.	PARTIC.	0.757	1.856	1.389	1.815	1.439
		1.409	3.171	2.409	2.754	2.453
		-0.380	-1.177	-1.223	-1.210	-1.064
AMOUNT UNDER CURRENT LAW		1488336.	1757479.	3063560.	1156312.	7465687.
		3759733.	4606453.	8370584.	3270035.	20006805.
		217349045.	308745528.	542551534.	185784578.	1254430685.

HOUSEHOLDS THAT WERE SOLELY SSI IN SS CASHOUT STATE = 0

Table of Contents

APPENDIX C

NEW SOFTWARE TOOLS: CASE AND OOP

NEW SOFTWARE TOOLS: CASE AND OOP

CASE technology encompasses all aspects of the software development life-cycle, ranging from project management, data and systems analysis, system design, automated code generation, and maintenance. The technology even goes one step further by offering re-engineering tools, which allow you to take a program already written and put it into the CASE environment after the fact. This technology was introduced in the 1970s, but it still has a way to go before most corporations fully embrace it. Dr. Rubin of Howard Rubin Associates reports that even though 70 percent of U.S. companies have front-end CASE tools, only 10 percent really use them (Williamson, 1992 c). The reason for such a delayed embrace of CASE technology can be summarized by Sprague's comments (1992):

> While many CASE tools can be effective, the real question is whether organizations are ready to use them: few are. Typically, organizations focus solely on the tool as a panacea. This results in unmet expectations, frustration and an inability of the I/S [Information System] function to deliver value. I/S is likely to be less productive because the tool distracts developers from the actual delivery process. As has been said, "A fool with a tool is a damned fool."

We must admit that we experienced Sprague's comments. We sought to discover whether CASE tools could prove to be an effective design tool for us. We purchased one called Easy CASE Plus to see for ourselves how useful the package could be. We found that it was helpful in graphically depicting the data flow of the Department of Labor's FEDS Information database *after* the design was made. We found that it would not be as useful *during* the design phase, since it was not flexible enough. We found that the tool would have prolonged the design phase rather than expedited it. Another problem with such tools is that the user must know not only the methodology well (whether it is Gane-Sarson or Coad-Yourdan), but must also have graphic expertise. We found the whole process to be just too labor intensive.

Typically, the firms that have successfully implemented CASE technology are developing more traditional corporate decision support applications. These projects generally require more developers on the project team and offer more repeated CASE experiences (Rinaldi and Gannon, 1992), which at this time would not be feasible for us. This will probably change in the next few years as the U.S. Department of Defense, the largest software customer in the U.S., acquires a complete software engineering environment composed of off-the-shelf CASE tools (Williamson, 1992b). This procurement will have a profound impact on the entire CASE industry, from which we are sure to benefit. We believe we should wait for CASE to evolve into a more flexible, easy-to-use package before applying CASE tools for microsimulation model design. We also need to be sure that microsimulation can truly benefit from them.

We are much less likely to benefit from the OOP technology now or in the future than from CASE technology. Many people misunderstand OOP as being a specific software package, when in fact it really is a programming methodology. Conventional programming languages like Pascal, FORTRAN, and BASIC have been used in OOP, albeit not very easily (Faison, 1992). Languages like Smalltalk, Lisp, Eiffel, and nowadays C++ are more commonly used to implement OOP.

What is all the excitement surrounding OOP? The excitement began in the 1970s with the rising popularity of OOP among programming language researchers (the concept of artificial intelligence programming was gaining popularity at that time, too). The design of the OOP methodology focuses on providing programmers with the means to develop and maintain more powerful and independent program units[1] that can be easily reused and adapted to a new scenario. These program units are more than just functions, in the usual programming sense. They really are objects, because they contain both the data structure and the functions that act upon that data. These objects usually represent a physical or an abstract entity. Some examples of objects currently in use today are Integer, Array, and Stack. Some examples in the graphical users interface (GUI) environment are

[1]Program units does not imply programs or functions, per se.

TRadioButton, TCheckBox, and TButton. Some examples in the database environment are TBrowse and TEdit.

OOP allows the programmer to focus on the *relationships* between the objects, rather than the *implementation* details. The big payback of OOP is that it is considered easier and less costly to maintain; its foundation centers on reusability. For instance, the user may want to add a new capability to an object instead of following the usual tedious course of reviewing a specification, modifying the current implementation, unit testing the change, system testing it, and hoping that some other part of the system will not be affected by it. The user then simply inherits the object, adds the new capability to the object, and starts using the new object instead of the old one. The original object does not need to change. For example, the user may have designed a box to display on the screen as part of the GUI, and now wants to make the box push-button. The new push-button object can easily be created by incorporating or inheriting the functionality of the box, not the implementation of the box. All the push-button object needs to know are what functions it can use and what data is accessible. Again, the focus is on the relationship among the objects, not the implementation.

We should not use the OOP methodology for transferring the MATH model to the PC even if we redesigned and rewrote the model. The purpose of the MATH model is to simulate tax and transfer programs using data from the CPS. Simply stated, the MATH model reads data, processes data, and writes data. There are not many opportunities to define an object (data and related functions) and reuse it. We tried to implement this approach, but it proved too difficult to follow and maintain.

The only time we might consider using the OOP methodology is when we design the user interface for the MATH model. Already the CLIPPER programming language is based on OOP and encourages the use of its TBrowse and TEdit objects. As for other aspects of the model, we recommend against using OOP.

Table of Contents

APPENDIX D

UNIX-BASED WORKSTATION TECHNOLOGY

UNIX-BASED WORKSTATION TECHNOLOGY

UNIX is an operating system; however, when we discuss UNIX workstations, we are talking about a non-Intel-based hardware platform that uses UNIX as its operating system. Even though we have projected that the PC platform will be sufficient for executing the MATH/PC model, we believe a brief review of the UNIX-based platform is still pertinent, especially since more and more agencies of the federal government are moving towards UNIX-based platforms for some types of environments. We are not sure why this is happening, since there is no requirement that agencies replace PCs with UNIX workstations. However, the federal government is strongly encouraging the development of open systems, which today are becoming possible in the UNIX arena. An open system is not tied to a specific hardware or software vendor. Any machine can communicate with any other machine using any type of operating system. We believe that the government's efforts could strongly influence computing capabilities at FNS and at MPR. With this in mind, we begin this disucssion with a brief description of UNIX and its capabilities and then review the differences between the PC- and UNIX-based platforms.

A. UNIX-BASED PLATFORM

A UNIX-based platform consists of a machine using UNIX as its operating system. The machine could be a PC, based on an Intel microprocessor, a mainframe, or a workstation. A UNIX workstation is usually a single-user machine, whereas UNIX is in general a multi-user OS. UNIX is capable of using the full extent of the powers of a workstation, whether it be a PC or a non-Intel platform.

The evolution of the UNIX operating system explains why there are so many variations of UNIX in operation today and why the UNIX market has catered to academic and scientific research communities instead of mainstream corporate businesses. The UNIX operating system began in the late 1960s at AT&T Bell Laboratories, with Ken Thompson and Dennis Ritchie (Chan, 1992). Their

mission was to create a system for editing and formatting text, which was a formidable task at that time. They completed their mission with an operating system that ran on a PDP-11/20 with 24KB of memory and 0.5MB of disk space. The cost was $65,000. By the mid-1970s, they rewrote the UNIX kernel in the C programming language and started licensing it to Western Electric for $200. In 1981 they had UNIX system III. Around the same time, the University of California at Berkeley developed their own version of UNIX. And in the mid-1980s, AT&T tried to make its new version, System V, the standard. From then on, companies such as Sun Microsystems, Hewlett Packard, IBM, Digital, and Santa Cruz Operation created their versions of UNIX based on either Berkeley or AT&T UNIX (Fiedler, 1992). The result is an operating environment in which the operating systems are very similar. Only minor functions or capabilities keep them distinct.

Berkeley and AT&T tailored UNIX toward academic and scientific research institutions with features that put it in a different class from a DOS or mainframe operating environment. The result is an industrial-strength operating system that enables multiple users, multiple tasks, and very large programs to work at the same time efficiently. UNIX was designed to be as powerful as a mainframe OS, but with lower costs, and as easy to use as DOS, but with more sophisticated features.

In a sense, the sophistication of the operating system parallels the mainframe operating environment (multiuser, multitasking). UNIX is easier to manage and use than MVS (IBM's chief mainframe operation system) since it does not manage resources as much as MVS. At a minimum, one system administrator is needed to set up user accounts, maintain adequate resource sharing, and install system upgrades. For someone who is comfortable with DOS, this environment can be quite intimidating. But for people accustomed to working on a mainframe and aware of system operations, it is not as intimidating. Using job control language or scheduling jobs are not concerns. A program does not have to be run at night to take advantage of cost discounts. This is especially appealing for us, given the required timeliness of the reform simulations. Of course, all of these benefits come at the cost of training people to be proficient system administrators.

From the start, engineers and technical programmers have enjoyed the power and flexibility of UNIX. In general, UNIX has always had a wider availability of scientific and engineering-related applications, not usually found on other operating systems. For instance, the interactive UNIX environment is a clear favorite for engineers wanting to model fluid dynamics. For technical programmers, UNIX provides an operating system that the programmer can control. If the programmer does not like the way the operating system works, the programmer can change it by writing a C program that communicates with the system. In addition, the programmer can actually modify the program code of the operating system. However, for most corporate data processing groups, the power and versatility of UNIX is not as enticing. The batch mainframe environment is sufficient for their accounting, purchasing, and inventory needs, especially since UNIX did not offer many packages of that kind.

Only recently has the market share of the UNIX arena become large enough for makers of word processing, spreadsheets, databases, and other applications to see a need or incentive to develop their packages for that operating system. UNIX now provides those packages and has evolved into more than just an operating system for the scientific and engineering world. It has become another strong contender for persons deciding to use DOS or OS/2 environments for their data processing needs.

B. COMPARISON OF UNIX AND PC PLATFORMS

A comparison of UNIX and PC operating environments shows a UNIX-based platform to be more powerful than the PC-based platform, but it would require more resources to purchase, train personnel, and maintain than a PC-based platform. As Jim Seymour (1992) states:

> Temptation abounds. You want the power of a Sun workstation, the ease of a Macintosh, the cool elegance of a NeXTstation. But when it comes to getting your work done, you come back to the PC.

The UNIX alternative only makes sense if the computing power is required or if the strategic goals of the corporation are to harness the power and versatility of the UNIX platform. It is still too expensive an option in which the curious can dabble.

In Table D.1, we show a comparison between DOS, OS/2, and UNIX.[1] UNIX can handle multiple users, whereas neither of the other operating systems can. Both UNIX and OS/2 support multitasking, for example, running the MATH/PC model and the QC Minimodel at the same time. However, UNIX has had multitasking for more than 15 years, whereas OS/2 has been on the market for less than five years. UNIX is supposed to be vendor independent, but not all variations of UNIX run on all platforms. DOS and OS/2 require IBM or IBM-compatible computers, of which there are dozens if not hundreds to choose from. While OS/2 has memory capabilities that are far superior to those of DOS, UNIX surpasses both; it is limited only by the hardware's limitations. If UNIX is running on an Intel 80386 or 80486 machine, the limit is 4.3 billion bytes. This means that programs do not have to be designed to execute under a certain size in memory and constraints on how much data can be manipulated in memory is not a concern.

In Table D.2, we show comparisons between the two popular Windows packages: Windows, which runs under DOS, and X-Windows, which usually runs under UNIX. The Windows interface offers a pseudo-multitasking environment. Windows appears to be multitasking to the user, but in actuality it is multitasking only to the extent that the applications designed to run under Windows allow it to be multitasking. The applications control when to take a break and check with the operating system to see if any other tasks need resources. If the application never checks with the operating system, the application will never multitask with another application. X-Windows, on the other hand, is a true multitasking interface. The operating system, not the application, controls the multitasking process. Whether an application likes it or not, UNIX will check at various intervals to see if other tasks need resources. Another drawback of Windows is that it is the property of the

[1]We exclude Windows from this discussion because it is not a separate operating system; it works under Microsoft's DOS.

TABLE D.1

COMPARISON OF DOS, OS/2, AND UNIX OPERATING SYSTEMS

	DOS	OS/2	UNIX
Number of Users	1	1	Multiple
Number of Concurrent Tasks	1	Multiple	Multiple
Vendor-Specific	Intel Chips	IBM-PC, compatibles 386 or better	Yes
Memory (RAM) Limitations	640K of RAM in a 1 MB address space	16MB of RAM	Hardware Dependent (on Intel 80386, 80486 - 4.3GB)
Year Developed	1981	1987	1969

SOURCE: Uniprime Systems, Inc. (1992)

TABLE D.2

COMPARISON OF MICROSOFT WINDOWS AND X-WINDOWS

	Windows	X-Windows
Operating System	Microsoft DOS	Almost Any
True Multitasking	No	Yes
Hardware Specific	Intel Platform	Almost Any

SOURCE: Uniprime Systems, Inc. (1992)

Microsoft Corporation. It can run only under Microsoft DOS; no other PC-DOS can run it. Also, it requires an Intel-based platform. X-Windows, a public domain software package, can run on virtually any computer, under almost any operating system. Although it is primarily used under UNIX, it is not a UNIX product.

In Table D.3, we show how the UNIX operating system includes more than just an operating system in its kernel. Features such as a C compiler, network interface, tape drive interface, and change control software are standard with UNIX. In the DOS and OS/2 environments, each one would be purchased separately, and the user would have to pay special attention to compatibility among products.

C. STRENGTHS AND WEAKNESSES

It is clear from these comparisons that UNIX provides a robust, mature, and capable operating system. It would probably be faster for some types of application (the MATH model being one) and worse for others. Workstations tend to be very efficient at floating-point operations compared to PCs, but their integer and other types of operations are not significantly different (see Table V.3). As mentioned earlier, most corporations have not embraced the UNIX operating environment because of its complexity and lack of software packages. Most corporations do not need specialized scientific or engineering applications. UNIX workstations, especially high-end ones, have substantially more computing power, but the cost goes up significantly for hardware, training, and software. We are tempted, because of the power of UNIX, to apply it to the MATH/PC model. If our predictions about how fast the MATH/PC model executes under DOS are wrong by a factor of two, three, or more, then UNIX would be the best alternative operating system, unless of course we revert back to the mainframe.

The main drawbacks of UNIX are its complexity and its price. The complexity derives from its multiuser, multitasking nature. It would be impossible to have an operating system as simple as DOS to perform those functions. Once a system administrator is well-trained and experienced in UNIX,

TABLE D.3

COMPARISON OF FEATURES THAT ARE INCLUDED IN/EXCLUDED FROM DOS, OS/2, AND UNIX

Feature	DOS	OS/2	UNIX
Operating System	Included	Included	Included
Editor	Included	Included	Included
FORTRAN Compiler	Excluded	Excluded	Excluded
C Compiler	Excluded	Excluded	Included
COBOL Compiler	Excluded	Excluded	Excluded
Library Facility	Excluded	Excluded	Included
Backup/Restore	Included	Included	Included
Tape Drive Interface (???)	Excluded	Excluded	Included
Networking	Excluded	Excluded	Included
Change Control	Excluded	Excluded	Included
Absolute Debugger	Excluded	Excluded	Included
Symbolic Debugger	Excluded	Excluded	Included
Profiler	Excluded	Excluded	Included
Assembler	Excluded	Excluded	Included
Word Processor	Excluded	Included	Excluded
Spreadsheet	Excluded	Included	Excluded
Online Documentation	Included	Included	Included
Base Price	$75.00	$99.00	$2,000.00

and once the users of UNIX, who are currently very experienced in DOS, learn the new operating system commands, the real work could begin. We believe that the training would require many more resources and more time than are currently available for this project. Also, we believe that UNIX may have a place in the future for FNS-sponsored work and perhaps even for the MATH/PC model. We suggest a wait-and-see approach rather than trying to use it now.